Principia Data

By Ismaël Chang Ghalimi

Principia Data received major contributions from Christian Kaas, Florian Ribon, and François Beaufils.

Many thanks to them.

Work on *Principia Data* was funded by Sutoiku, Inc.

Many thanks to their customers and investors.

Questions and comments should be sent to ismael@stoic.com.

data

: factual information (such as measurements or statistics) used as a basis for reasoning, discussion, or calculation.

Source: Merriam-Webster

Principia Data is a unified typology of statistical variables. Its purpose is to facilitate the collection and analysis of structured datasets by typing their variables with as much precision as possible, while remaining domain agnostic. *Principia Data* draws its inspiration from the fields of computer science and statistics, but was not vetted by any formal peer review process.

Table of Contents

Audience

This document is aimed at anyone interested in learning more about the fundamental nature and structure of data, for the purpose of improving one's understanding of common datasets and one's ability to analyze and visualize them properly.

Formal training in the disciplines of computer science or statistics are not required *a priori*, and could in fact prevent a careless reader from keeping an open mind about the subject at hand. Nevertheless, the reader is strongly encouraged to refer to other reference materials whenever some notions deserve deeper investigations.

Background

One of the original taxonomies for levels of measurement was proposed by the psychophysicist Stanley Smith Stevens, who defined **nominal**, **ordinal**, **interval**, and **ratio** scales in a 1946 Science article titled "On the theory of scales of measurement". Mosteller and Tukey (1977)[1] distinguished **grades**, **ranks**, **counted fractions**, **counts**, **amounts**, and **balances**. Nelder (1990)[2] described **continuous counts**, **continuous ratios**, **count ratios**, and **categorical** modes of data.

Unfortunately, none of these taxonomies or typologies for statistical data types provide a sufficient model for describing the complexity of common datasets found across a broad range of applications and domains. More specifically, *Principia Data* was designed to address the following requirements in a coherent manner:

- Establish a clear distinction between discrete and continuous variables;
- Establish clear discretization paths between continuous and discrete typologies;
- Organize typologies in an incremental fashion through an algebraic topology;
- Capture time in a comprehensive manner (discrete, continuous, directional);
- Provide proper support for absolute variables;
- Provide proper support for directional variables (*Cf.* Directional Statistics);
- Provide proper support for identificational variables (*e.g.* unique key);
- Provide proper support for relational variables (*Cf.* Graph theory);
- Provide proper support for geospatial variables (both positions and geometries);
- Establish a clear distinction between dependent and independent variables.

[1] Mosteller, F., & Tukey, J. W. (1977). Data analysis and regression. Boston: Addison-Wesley.
[2] Nelder, J. A. (1990). The knowledge needed to computerise the analysis and interpretation of statistical information. In Expert systems and artificial intelligence: the need for information about data. Library Association Report, London, March, 23–27.

Variables

The Australian Bureau of Statistics defines a variable as *any characteristics, number, or quantity that can be measured or counted*. A database administrator could define it as anything that would fit into a column of a table within a relational database management system. Similarly, a business analyst could define it as anything that would fit into a column of a sheet within a spreadsheet workbook, while preserving a common type and format across all rows.

All these definitions are acceptable for the purpose of *Principia Data*.

Discrete *vs.* Continuous

Principia Data is designed upon a fundamental dichotomy between **discrete** variables and **continuous** variables. A discrete variable can be represented or encoded using an integer, while a continuous variable is represented or encoded using a real number.

For a discrete variable, *encoded* means that values of the variable can be captured as integers in conjunction with some kind of lookup table that associates the variable's actual values to their corresponding integers used as indexes. For example:

Index	**Value**
1	red
2	green
3	blue

The distinction between discrete variables and continuous variables is usually quite well understood by statisticians, who define discrete probability distributions for the former, and continuous probability distributions for the latter.

This distinction is also essential to the analysis of multivariate datasets (datasets made of multiple variables), for the dimensions of an aggregation (a.k.a. pivot) must all be discrete, while its measures could be discrete or continuous (*Cf.* Independent *vs.* Dependent).

Finally, from an information theory standpoint, the encoding of discrete variables is usually more space efficient than the encoding of their continuous counterparts.

Basic Continuous Variables

The modern field of statistics, which emerged in the late 19th and early 20th century, was mostly concerned with continuous measures. The statistical analysis of discrete measures was largely developed in the second half of the 20th century. Following this chronology, we will start our categorization of statistical variables with continuous ones.

In the continuous domain, the physical properties of materials and systems can often be categorized as being either intensive or extensive quantities, according to how the property changes when the size (or extent) of the system changes. According to IUPAC, an intensive property is one whose magnitude is independent of the size of the system. An extensive property is one whose magnitude is additive for subsystems. The terms intensive and extensive quantities were introduced by Richard C. Tolman in 1917[3]. [Source: Wikipedia]

Nevertheless, the original taxonomy introduced by Stanley Smith Stevens in 1946 considered only one type of continuous variables called **ratio**. Only in the 1960's did statisticians start to distinguish between extensive variables that can be summed, and intensive ones that cannot. Furthermore, this distinction was made for continuous variables only. Building on this legacy, *Principia Data* adopts the same terminology.

In what follows, every typology is defined with an alias using common words of the English dictionary in order to make typologies more approachable by a less technical audience. Whenever possible, the typology's name is the adjective corresponding to its alias.

[3] Tolman, Richard C. (1917). "The Measurable Quantities of Physics". Phys. Rev. 9 (3): 237–253.

Intensive Variables

Alias: *Level*

Family: Continuous

Example: 10°C, 20°C, 30°C

A continuous variable is said to be **intensive** if its values cannot be summed. Examples of intensive variables include speeds, altitudes, or product prices. The fact that two speeds or two altitudes cannot be summed is pretty obvious, but the non-summability of two product prices might deserve more explanations.

When going to the grocery store, the grand total displayed at the cash register is not a direct sum of product prices. Instead, it is the sum of individual product prices each multiplied by product counts (*Cf.* Cardinal Variables) or extensive quantities (*Cf.* Extensive Variables), which themselves are summable (*Cf.* Arithmetic Relationships).

By definition, all physical intensive quantities are intensive.

Extensive Variables

Alias: *Amount*

Family: Continuous

Example: 10m^2, 100m^2, 1000m^2

A continuous variable is said to be **extensive** if its values can be summed. Examples of extensive variables include weights, areas, or account balances. The progression from *intensive* typology to *extensive* typology is therefore enabled by the injection of a mathematical operator (*summation*). This incremental refinement of typologies through successive mathematical operators is a fundamental design principle of *Principia Data* and will be referred to as *algebraic topology* through the remainder of this paper.

By definition, all physical extensive quantities are extensive.

Arithmetic Relationships

The difference between two values of an intensive variable is itself an extensive variable. For example, the difference between two altitudes is a climb, and values of a climb can be summed.

Similarly, the difference between two values of an intensive variable can be added to a value of the same intensive variable, or to the value of another intensive variable expressed in the same unit of measurement. For example, a climb can be added to an altitude.

Furthermore, the product of an extensive variable (*e.g. gallons of gasoline*) by an intensive one (*e.g. price of gasoline per gallon*) makes an extensive variable (*e.g. price of a gas tank fill-up*). Similarly, the product of two extensive variables makes an extensive variable, while the product of two intensive variables makes an intensive variable.

Variable 1		**Variable 2**		**Product**
intensive	×	*intensive*	=	*intensive*
intensive	×	*extensive*	=	*extensive*
extensive	×	*extensive*	=	*extensive*

Non-Summability of Variables

What makes some continuous variables non summable, therefore intensive? For example, why do a car driving at *25 km/h* and another driving at *50 km/h* do not make one driving at *75 km/h*? The arithmetic answer is often related to the fact that many intensive variables are defined as the result of arithmetic divisions, and the results of divisions cannot be summed freely, in much the same way that averages cannot be summed.

Nevertheless, the non-summability of a variable is not necessarily an invariant of the formula with which it is computed or its unit of measurement. Instead, it is a matter of context, which makes the intensive *vs.* extensive dichotomy all the more subtle. For example, consider the speeds of cars competing in a race. Speed, being the result of dividing a distance by a duration is clearly an intensive variable, and therefore cannot be summed. Or could it?

Well, it turns out that one could sum these speeds, but the result of this summation would not represent the speed of any car. It would represent the speed at which aggregated distances are being traversed by cars during the race. Whether this information is meaningful or not is a question to be asked to physicists or race car drivers — it is not for *Principia Data* to answer it. This aggregated speed can itself be defined as a variable, and being the result of a summation, it is defined as an extensive variable, for the arithmetic *summation* operator is associative.

This example illustrates a fundamental aspect of the non-summability of intensive variables and their discrete counterparts (*Cf.* Incremental Variables): the summation operator used to distinguish extensive variables from intensive ones is an operator which result computes values of a variable that is *congruent* with the variable being summed. Congruence in this context remains loosely defined, but we could approach it with the following questions:

> *Could results of the summation represent legitimate values of the summed variable?*

In the previous example, the valid answer is clearly *"no"*, but let us consider a use case that could lead to a positive answer. For example, we could use the heights of elevators in buildings. Buildings themselves cannot be stacked, therefore their heights should be defined as an intensive variable. But the heights of their elevators is directly related to the lengths of elevator cables, and these could come from the same spool and be used to construct a taller elevator. Therefore, from the viewpoint of an elevator manufacturer, the heights of elevators could be legitimately defined as an extensive variable. Nevertheless, a skilled data architect might prefer to consider the *Elevator Cable Length* variable, which extensive nature would be less ambiguous.

Following this example, it should be noted that not all intensive variables are the results of divisions. Furthermore, the fact that an intensive variable is the result of a division can be checked by looking at the unit of measurement of the variable, as expressed using base units. The variable is the result of a division if and only if its unit of measurement includes a division.

Basic Discrete Variables

In order to support the discretization of any continuous variable (*Cf.* Variable Discretization), *Principia Data* defines a discrete dual for each and every continuous typology. The discrete dual of the intensive typology is called **incremental**, while the discrete dual of the extensive typology is called **cardinal**. These terms have been used inconsistently throughout various disciplines, therefore should be used carefully, especially when comparing *Principia Data* to other taxonomies or typologies for statistical data types. It should also be noted that not all discrete typologies have a continuous dual (*Cf.* Summary of Basic Typologies).

Incremental Variables

Alias: *Increment*

Family: Discrete

Example: 10, 20, 30

A discrete variable is said to be **incremental** if its values cannot be summed. An example of incremental variable is the height of a building measured in full meter increments, for buildings cannot be stacked on top of each other (*Cf.* Non-Summability of Variables).

Cardinal Variables

Alias: *Count*

Family: Discrete

Example: 10, 100, 1000

A discrete variable is said to be **cardinal** if its values can be summed. An example of cardinal variable is the population of countries, which can be summed by regions like continents.

Starting from the Beginning

This simple distinction between summable and non-summable variables in the discrete domain was the original insight that lead to the development of *Principia Data*. Nevertheless, the incremental and cardinal typologies just introduced are quite advanced with respect to the set of mathematical operators they support, and cannot be properly understood without starting from much simpler ones. Therefore, we shall start with the simplest of all: the boolean typology. From there, we shall build our way back up to the incremental and cardinal typologies, thereby establishing *Principia Data*'s core algebraic topology.

Boolean Variables

Alias: *Boolean*

Family: Discrete

Example: TRUE, FALSE

The most primitive variable that can be defined is one that can only take two discrete values. Some statisticians call it a binary variable. In reference to boolean algebra, *Principia Data* prefers the term **boolean** variable. Examples of boolean variables include true/false statements, yes/no answers, or obsolete forms of gender discrimination.

Nominal Variables

Alias: *String*

Family: Discrete

Example: Foo, Bar, Baz

In order to expand from the strictly binary scope of a boolean variable, one must add a simple mathematical operator we call *extension*. This operator lets one extend the set of possible values for the variable beyond just two. A discrete variable with three or more possible values is called **nominal**, in reference to the original taxonomy introduced by Stanley Smith Stevens. Alternative taxonomies or typologies sometime use the term *categorical*, but the term *nominal* was prefered for its ontological proximity to the next discrete typology (*Cf.* Lexical Variable). Examples of nominal variables include the names of objects or the colors of crayons. It should be noted that values of nominal variables cannot be sorted, for the nominal typology is not defined with any ordering relation (*Cf.* Order theory).

Lexical Variables

Alias: *Word*

Family: Discrete

Example: Alpha, Bravo, Charlie

A nominal variable defined with an ordering relation is called a **lexical** variable. Examples of lexical variables include the names of people or places, the Alpha 3 Codes of countries, or the alphabetical letters identifying rows of seats in a stadium.

The term *lexical* was prefered to *alphabetical*, because the ordering relation defined for sorting can be lexicographic, not just alphabetic. In fact, any ordering relations, not just lexicographic ones, could be used to define lexical variables.

It should be noted that not all variables represented with lexicographic symbols can be sorted. For example, the names of objects like *bicycle* or *car* have different translations in different languages, with different lexicographic representations. Therefore, any ordering relation defined for them would be purely arbitrary and devoid of any signification. Values of such a variable expressed in a common language could be sorted when displayed or printed, but this sorting would be used for convenience purposes only. In the abstract context of the nominal variable at hand, its values are not sorted in any way.

In contrast, the names of places expressed in their predominant local language and encoded using Unicode characters can be sorted in a slightly more legitimate fashion, even though the standard ordering of Unicode characters across its 139 modern and historic scripts is itself totally arbitrary. But the alphabetic ordering of rows of seats in a stadium is not, for row *A* precedes row *B* and row *B* precedes row *C* in the physical space of the stadium.

One might wonder why it should matter that the values of certain variables can be sorted, while the values of others cannot. This is a valid question indeed, which deserves some illustration. The primary motivation for establishing such a distinction is to prevent the introduction of any bias when analyzing a dataset, especially when this dataset is being visualized.

Visualizing a dataset consists in projecting an n-dimensional topological structure onto the bidimensional structure of a sheet of paper or computer display. A holographic display might add a third dimension across space, and animations a fourth one across time. But whichever rendering apparatus is employed, its multidimensional structure is essentially cartesian, in the sense that each of its dimensions (*X*, *Y*, *Z*, *t*) is defined with an ordering relation.
As a result, when projecting a dataset which variables might be boolean or nominal (unordered) onto the cartesian structure of a visualization, a bias is automatically introduced. Therefore, the proper typing of variables as nominal ones (versus lexical ones) is aimed at reducing the misleading impact of this intrinsic bias with the help of various visualization techniques.

For example, a choropleth map could be used to visualize all countries with their respective predominant religions depicted with different colors. Such a visualization would ask the artist to answer two important questions:

1. Which color palette should be used to visually distinguish the different religions?
2. In which order should the religions be listed on the map’s legend?

These questions are important, because the artist will want to ensure that no ordering relation is implied by the color palette and the legend ordering, so as not to offend the members of any religious group. Knowing that the predominant religion of a country is a nominal variable instead of a lexical one will automatically provide the following answers:

1. Use a nominal (a.k.a. qualitative) color palette instead of a sequential one.
2. Use a random ordering for the legend.

Sequential Variables

Alias: *Sequence*

Family: Discrete

Example: 1st, 2nd, 3rd

So far, values of the basic discrete typologies that have been introduced according to our algebraic topology (boolean, nominal, lexical) have been represented with lexicographic symbols instead of numerical ones. Granted, all these values could be encoded using integers (*Cf.* Discrete *vs.* Continuous), but this numerical encoding does not confer any numerical properties to the variables at hand, especially with respect to the availability of arithmetic operators that could be used to analyze these variables (they positively cannot).

Sometimes though, a discrete variable defined with an ordering relation could be legitimately represented with an integer (signed or unsigned). In such a case, the variable is called **sequential**, for it can be represented as an ordered sequence of integers. Examples of sequential variables include the arrival ranks of contestants in a race (*1st, 2nd, 3rd, etc.*) or the ranks of countries sorted by decreasing count of population. In alternative taxonomies or typologies, this typology is sometimes called *rank*.

Sequential variables differ from lexical ones in a fundamental way: the ordering relation of sequential variables is unique and unambiguous, and is defined by the ordering relation of natural numbers ($\mathbb{N}$) or signed integers ($\mathbb{Z}$). Furthermore, integers are fully sufficient to represent the values of sequential variables, while lexical variables usually require a lookup table (*Cf.* Discrete *vs.* Continuous) when they are encoded using integers.

Nevertheless, representing the values of a sequential variable with integers does not confer it the arithmetic operators like summation or subtraction usually attached to this set of numbers.

To convince oneself of this important limitation, one could consider again the arrival ranks of contestants in an Olympic race. Taking subtraction as an example (subtraction precedes summation in the algebraic topology defined in *Principia Data*), one could consider that one rank separates the *2nd* from the *1st*, while one rank also separates the *4th* from the *3rd*. But the single rank of separation from the *4th* to the *3rd* does not hold the same signification as the rank of separation from the *2nd* to the *1st*, for the *4th* contestant, unlike the preceding three, does not get a medal and does not step on the podium. And only the gold medalist gets an anthem.

Ordinal Variables

Alias: *Index*

Family: Discrete

Example: 1, 2, 3

The first arithmetic operator in the algebraic topology defined in *Principia Data* is the *subtraction*. The precedence of subtraction over summation is motivated by the fact that it is already present in the traditional dichotomy between intensive and extensive variables, since both intensive and extensive variables support the subtraction operator.

A discrete variable which values can be represented as integers and can be subtracted from each other is called an **ordinal** variable. Once again, this term was used inconsistently by alternative taxonomies and typologies and should be used with caution when comparing them to the typologies defined by *Principia Data*. Examples of ordinal variables include the lengths of words expressed in character counts with a given lexicographic system, the number of timezones in use across the geographical expanse of a country, or the number of populated values for individual columns within a database table.

While the term *ordinal* is rather common in the scientific literature covering the broad field of levels of measurement, in the context of *Principia Data*, it designates a typology that is rather rarely found with common variables. This is due to the fact that ordinal variables do not support the summation operator, while most conventional discrete variables that can be represented using integers usually support this operator, at the exception of ranks, which belong to the sequential typology of variables. In other words, a discrete variable supporting subtraction but not summation is not easy to come by.

Back to Incremental Variables

Finally, we can go from ordinal variables to incremental variables (*Cf.* Incremental Variables) by adding the *subdivision* operator. This mathematical operator should not be confused with the traditional division operator, for we are not dividing values of incremental variables by themselves. Instead, we are subdividing such values by values of other variables or constants that are defined externally. An example of an incremental variable could be the height of buildings measured in full meters. Indeed, every *meter* can be subdivided into *100 centimeters*, and the height of the same buildings can be expressed in centimeters.

Following such a definition, one might wonder which ordinal variables do not support such a subdivision. The answer becomes quite clear when one considers variables related to counts of indivisible entities such as people. An example of such an ordinal variable would be the subset of occupants inside an elevator who are over the age of *18*.

Summary of Basic Typologies

The basic typologies defined beforehand can be summarized as follows:

Operator		**Extension**	**Sorting**	**Sequencing**	**Subtraction**	**Subdivision**	**Summation**
Discrete	*Boolean*	*Nominal*	*Lexical*	*Sequential*	*Ordinal*	*Incremental*	*Cardinal*
	TRUE	Red	Alpha	1st	1	10	10
	FALSE	Green	Bravo	2nd	2	20	100
		Blue	Charlie	3rd	3	30	1,000
Continuous						*Intensive*	*Extensive*
						10°C	$10m^2$
						20°C	$100m^2$
						30°C	$1,000m^2$

At a glance, one will notice that *Principia Data* defines many more discrete typologies than continuous one. There are multiple reasons for this. First, traditional typologies like the one defined by Stanley Smith Stevens had a similar imbalance (3 *vs.* 1). Second, and maybe most importantly, while defining continuous duals for discrete typologies such as sequential or ordinal could be done, it would offer limited benefits, because losing the subdivision operator would establish a perfect bijection between dual sets.

Nevertheless, the author recognizes that some basic continuous typologies might still be missing, starting with a direct successor to the extensive typology. Indeed, *Principia Data* in its current definition does not properly distinguish continuous variables that have an additive scale of relative difference and can be described with normal probability distributions, from continuous variables that have a multiplicative scale of relative difference and must be described with log-normal, gamma, or exponential probability distributions. To a very large extent, *Principia Data* remains a work in progress, and the present paper should be considered as a preliminary outline for a more rigorous academic exercise to be conducted in the future.

Nominal Sub-Typologies

Nominal variables are very common, and *Principia Data* provides a more detailed classification for them, built upon an algebraic topology orthogonal to the one used so far. This classification includes the following typologies:

Operator	**Typology**	**Example**
	Nominal	Foo, Bar, Baz
Textualization	*Textual*	Lorem ipsum dolor sit amet
Denomination	*Denominational*	John Doe
Identification	*Identificational*	stoic.com
Categorization	*Categorical*	Red, Green, Blue
Relation	*Relational*	USA$^{\times}$, Mexico$^{\times}$, Canada$^{\times}$
Hierarchization	*Hierarchical*	World$^{\times}$, North America$^{\times}$, USA$^{\times}$
Serialization	*Serial*	George Washington$^{\times}$, John Adams$^{\times}$

Note: the examples provided above for *relational*, *hierarchical*, and *serial* imply references to entities like *USA*$^{\times}$, *Mexico*$^{\times}$, and *Canada*$^{\times}$, instead of words like *"USA"*, *"Mexico"*, and *"Canada"*. This should become clear when reading the documentation for the *relational*, *hierarchical*, and *serial* typologies (*Cf.* Relational Variables). Relational references are denoted with the $\times$ symbol, in reference to the Crow's foot notation.

Textual Variables

Alias: *Text*

Family: Discrete

Example: Lorem ipsum dolor sit amet

A nominal variable which values can be represented as human-readable text is called **textual**. For example, the motto of a country. Examples of nominal variables that are not textual are binary blobs, encrypted strings, or computer software codes.

Denominational Variables

Alias: *Name*

Family: Discrete

Example: John Doe

A textual variable which values can be used as denomination for people, places, or things is called **denominational**. For example, the full name of a person, or the name of a capital city. This denomination is not free of homonyms though, therefore denominational variables should not be used for identification purposes.

Identificational Variables

Alias: *Identifier*

Family: Discrete

Example: stoic.com

A denominational variable which values can be used for the purpose of uniquely identifying people, places, or things is called **identificational**. For example, the email address of a person, the ISO 3166-1 alpha-3 code of a country, or the URL of a website.

Categorical Variables

Alias: *Category*

Family: Discrete

Example: Red, Green, Blue

An identificational variable which values can be used for categorization or classification purposes is called **categorical**. For example, the main religion of a country. The difference between an identificational variable and a categorical one is that all values of the latter can be enumerated, either *a priori* or *a posteriori*. In contrast, it is not possible to enumerate all possible email addresses, for their is an infinity of them.

Relational Variables

Alias: *Relation*

Family: Discrete

Example: USA×, Mexico×, Canada×

A categorical variable which values can be externalized into a separate dataset made of additional variables according to the relational model described by Edgar F. Codd is called **relational**. For example, the currency of a country, which can be defined in a separate dataset alongside its ISO 4217 code, fractional unit, and banknotes and coins denominations.

The decision to type a variable as categorical or relational is driven by several considerations:

First, if the variable must be described with additional metadata, it must be typed as relational.

Second, if the variable has a large cardinality (typically greater than 1,000), it should be typed as relational, for the possible values of categorical variables are usually treated as metadata, while the possible values of relational ones are usually treated as data, and the management of large sets of data are usually more scalable that the management of large sets of metadata when using conventional data analytics software packages.

Third, if the visualization of datasets is of primary importance to its users, a variable with a cardinality greater than 20 should be typed as relational. This is due to the fact that categorical variables will often be used for aggregation purposes, and the visualization of these aggregations might require the use of visual cohorts depicted with colored legends using nominal color palettes, which cannot be properly designed for more than 20 colors.

Hierarchical Variables

Alias: *Hierarchy*

Family: Discrete

Example: World×, North America×, USA×

A relational variable which references values of the same variable (reflexive relationship) is called **hierarchical**, for it defines a hierarchy across values of the variable itself. For example the associated nation of a country (*New Zealand* for *Cook Island*, *Italy* for the *Holy See*, *etc.*), or the manager in a list of employees. It should also be noted that in the example provided above for *World*×, *North America*×, and *USA*×, each of the three entities is a reference to a hierarchical region, not a country (*USA*× is a subregion of *North America*×, which is a subregion of *World*×).

Serial Variables

Alias: *Series*

Family: Discrete

Example: George Washington, John Adams, Thomas Jefferson

A hierarchical variable which parent values have no more than one child value is called serial, for it defines a set of series (*i.e.* linked lists). For example, the succession of heads of state for a country, or the successive arrivals of contestants in a race.

Nominal Variable Typing

The algebraic topology defined for nominal variables is aimed at supporting the proper typing of nominal variables. This is important, because when considering a new dataset, all variables should be considered as nominal by default.

This is due to the fact that the nominal typology is the most generic one (the one offering the lowest number of mathematical operators), at the exception of the boolean typology, but the latter enforces a constraint on cardinality, which cannot be assumed *a priori*.

This is also due to the fact that most computer systems use strings or binary objects for the interchange of datasets between applications. Therefore, the values of variables should be considered as strings by default.

From this point on, if a variable cannot be defined with mathematical operators like sorting, it should be typed as nominal first, then incrementally refined using the nominal typologies defined above. And when so doing, the most precise typology should be used, for it will help computer systems make the most appropriate decisions when dealing with values of the variable, either for computation or visualization purposes.

Obviously, this principle of precision should be applied to other typologies as well, going from nominal to lexical, from lexical to sequential, and so on and so forth, until the most precise typology has been selected for the variable.

Discrete Variable Cardinality

The *cardinality* of a discrete variable is defined as the number of individual values that can be taken by the variable. Sometimes, this cardinality is known in advance (*a priori*), as is the case for *boolean* or *categorical* variables (*Cf.* Categorical Variables). Sometimes, it is not, and new values are discovered later on (*a posteriori*). And some other times, the cardinality is potentially infinite, as can be the case for identificational variables (*Cf.* Identificational Variables).

Having a good understanding of the cardinalities of discrete variables and being able to estimate their orders of magnitude are essential to the task of selecting the most appropriate typologies for these variables. This is especially important for *categorical* and *relational* variables (*Cf.* Relational Variables), for such a selection can have dramatic impacts on the scalability and maintainability of applications developed on top of datasets incorporating these variables.

Numerical Typologies

Now that we have defined some basic typologies, we can construct more advanced ones through various specializations. To do so, we will focus on two sets of basic typologies: first, the set of **numerical** typologies, which includes the ordinal, incremental, and cardinal typologies in the discrete domain and the intensive and extensive typologies in the continuous domain, and will be used to define absolute, directional, epochal, and chronical typologies; second, the nominal typology, which will be used to define identificational and relational typologies.

Absolute Variables

Sometimes, the range of values for a numerical variable (*Cf.* Numerical Typologies) is bounded, either with a lower bound, an upper bound, or both. For example, temperatures expressed in kelvin (symbol: K) cannot be negative. Another example is the fraction of a subdivisible entity, which is between *0* and *1* (both included). *Principia Data* defines absolute duals for four of the five numerical typologies, ignoring the ordinal typology, for lack of sufficient use cases.

	Non-Summable	**Summable**
Discrete	Incremental → *Quantile*	Cardinal → *Fractional*
Continuous	Intensive → *Rational*	Extensive → *Percent*

Quantile Variables

Alias: *Quantile*

Family: Discrete

Example: 1st Quartile, 2nd Quartile, 3rd Quartile

An incremental variable which range of possible values is bounded is called **quantile**. For example, the values of a variable defining the quartile to which the area of a country belongs in relation to all other countries are integers between *1* and *4*. There is no fifth quartile, and quartiles cannot be summed.

Rational Variables

Alias: *Ratio*

Family: Continuous

Example: 0.25, 0.5, 0.75

An intensive variable which range of possible values is bounded is called **rational**. For example, the ratio of a country's area covered by water is greater or equal to *0* and strictly lower than *1* (a country cannot be entirely submerged, which will soon be a problem for some), but these ratios cannot be summed across countries.

Fractional Variables

Alias: *Fraction*

Family: Discrete

Example: ¼, ½, ¾

A cardinal variable which range of possible values is bounded is called **fractional**. For example, the ratio of a country's population in relation to the entire world's population and expressed in rounded-up per mille so as to make it a discrete variable (Symbol: ‰) is strictly greater than 0 (countries must have a non-null population to be recognized as such) and strictly lower than 1,000 (the most populous country holds less than a quarter of the world's population), and such fractions can be summed across regions like continents or signatories to international treaties.

Percent Variables

Alias: *Percent*

Family: Continuous

Example: 25%, 50%, 75%

An extensive variable which range of possible values is bounded is called **percent**. For example, the ratio of a country's area in relation to the area of the world's entire landmass is strictly greater than *0* (once again, a country cannot be entirely submerged) and strictly lower than *1* (the largest country represents less than 12% of the area of the world's entire landmass), and such ratios can be summed across regions.

Fractional vs. Percent

In the examples provided above, the one picked for fractional variables might seem a bit artificial. Indeed, the fraction of a country's population in relation to the entire world's population could have been expressed as a percent variable, which is the absolute dual of an extensive variable, which is itself a continuous variable. Nevertheless, a population count is a cardinal variable, hence of the discrete kind. Therefore, using a fractional typology for its fraction is more congruent to its discrete nature. And while this might seem like a very subtle difference, it could have significant consequences when visualizing datasets that include these variables, especially when conducting drill-down or drill-up interactions (*Cf.* Principia Pictura).

Directional Variables

Directional statistics (also [known as] *circular statistics* or *spherical statistics*) is the subdiscipline of statistics that deals with directions (unit vectors in $\mathbb{R}^n$), axes (lines through the origin in $\mathbb{R}^n$), or rotations (in $\mathbb{R}^n$). More generally, directional statistics deals with observations on compact Riemannian manifolds. [Source: Wikipedia]

The fact that 0 degrees and 360 degrees are identical angles, so that for example 180 degrees is not a sensible mean of 2 degrees and 358 degrees, provides one illustration that special statistical methods are required for the analysis of some types of data (in this case, angular data). Other examples of data that may be regarded as directional include statistics involving temporal periods (e.g. time of day, week, month, [...], *etc.*), compass directions, dihedral angles in molecules, orientations, rotations, and so on. [Source: Wikipedia]

In this context, *Principia Data* defines the following directional typologies as specializations of the absolute typologies defined earlier (*Cf.* Absolute Variables), through the introduction of the modulo mathematical operator:

	Non-Summable	**Summable**
Discrete	Quantile → *Meridional*	Fractional → *Sectorial*
Continuous	Rational → *Longitudinal*	Percent → *Angular*

Meridional Variables

Alias: *Meridian*

Family: Discrete

Example: 0°, 45° W, 90°E

The directional specialization of a quantile variable is called **meridional**. For example, the latitude of a capital city expressed in integer degrees.

Longitudinal Variables

Alias: *Longitude*

Family: Continuous

Example: 2° 20' 14.025" E

The directional specialization of a rational variable is called **longitudinal**. For example, the latitude of a capital city expressed in degrees, minutes, and seconds.

Sectorial Variables

Alias: *Sector*

Family: Discrete

Example: 10°, 20°, 30°

The directional specialization of a fractional variable is called **sectorial**. For example, the angle between the latitudes of a capital city and the closest capital rounded to the nearest degree.

Angular Variables

Alias: *Angle*

Family: Continuous

Example: 57.2958°

The directional specialization of a percent variable is called **angular**. For example, the angle between the latitudes of a capital city and the closest capital expressed in degrees.

Epochal Variables

Principia Data defines typologies for capturing instants or moments in time in relation to an epoch, defined as **epochal** specializations of the *ordinal*, *incremental*, and *intensive* typologies:

Subtractable Discrete	**Subdivisible Discrete**	**Subdivisible Continuous**
Ordinal → *Terminal*	Incremental → *Calendar*	Intensive → *Temporal*

Terminal Variables

Alias: *Term*

Family: Discrete

Example: 1st Term, 2nd Term, 3rd Term

The epochal specialization of an ordinal variable is called **terminal** (in reference to a *term*). For example, the first Summer Olympic Games to which a country participated since their refoundation in 1896 by Baron Pierre de Coubertin.

Calendar Variables

Alias: *Date*

Family: Discrete

Example: January 1, 1970

The epochal specialization of an incremental variable is called **calendar**. For example, a date, the hour of a date, or the month of a year. When encoded as a signed integer, *calendar* variables must be defined with a *periodicity*:

- *nanosecond*
- *microsecond*
- *millisecond*
- *second*
- *minute*
- *hour*
- *day*
- *week*[4]
- *fortnight*[5]
- *month*
- *quarter*
- *semester*
- *year*
- *decade*
- *century*
- *millennium*

[4] Culturally dependent, hence requires additional parameters.
[5] Culturally dependent, hence requires additional parameters.

Temporal Variables

Alias: *Timestamp*

Family: Continuous

Example: January 1, 1970, 00:00:00

The epochal specialization of an intensive variable is called **temporal**. For example, a timestamp (date and time, with time measured with as much precision as possible).

It should be noted that the dichotomy between *calendar* and *temporal* variables reflects the fundamental dichotomy between discrete time and continuous time. In such a context, the *calendar* typology can be considered as a discretization of the *temporal* typology, while the *terminal* typology can be considered as a re-discretization of the *calendar* typology.

Terminal *vs.* Calendar

One might wonder when the *terminal* typology should be used instead of the *calendar* one. Selecting the former instead of the latter follows the *principle of parsimony* (*Cf.* Occam's razor), which is prevalent in the field of statistics (and many others), and dictates that the simplest typology that could be used to capture all relevant properties of a variable should be selected, at the expense of more refined typologies (*Cf.* Precision *vs.* Parsimony).

The *terminal* typology can be considered as a re-discretization of the calendar *typology*. In this respect, the former is less precise than the latter. But what *terminal* lacks in precision, it gains in simplicity and terseness. Therefore, if an ordinal is sufficient to describe a term, it should be used as a more parsimonious alternative to a date or calendar period, which usually introduces a lot more complexity driven by the inherently cultural nature of calendaring systems.

Chronical Variables

Principia Data defines typologies for capturing instants or durations in time without any relation to an epoch, defined as **chronical** specializations of the *meridional*, *longitudinal*, *sectorial*, and *angular* directional typologies:

	Non-Summable	**Summable**
Discrete	Meridional → *Horological*	Sectorial → *Horometrical*
Continuous	Longitudinal → *Chronological*	Angular → *Chronometrical*

Horological Variables

Alias: *Moment*

Family: Discrete

Example: 2:56 AM

The chronical specialization of a meridional variable is called **horological**. For example, the first weekday in a country (*Sunday* in the United States of America, *Monday* in France).

When encoded as an integer, horological variables must be encoded with a pair of *periodicities*, the former being smaller than the later, and the value corresponding to the ordinal of the former period within the scope of the latter. For example:

- *second of minute*
- *minute of hour*
- *hour of day*
- *day of week*
- *day of month*
- *day of year*
- *week of month*[6]
- *week of year*[7]
- *month of year*

It should be noted that the *horological* typology is a chronical specialization of the *meridional* directional typology, because its values have upper and lower limits within the scope of a given periodicity. For example, the *hour of day* is between 0 and 24. There is no *25th* hour in a day.

[6] Culturally dependent, hence requires additional parameters.
[7] Culturally dependent, hence requires additional parameters.

Chronological Variables

Alias: *Time*

Family: Continuous

Example: 2:56:15 AM

The chronical specialization of a longitudinal variable is called **chronological**. For example, the timezone offset of a capital city or the departure time of a regularly scheduled flight.

Horometrical Variables

Alias: *Extent*

Family: Discrete

Example: 2:56

The chronical specialization of a sectorial variable is called **horometrical**. For example, the term length in years for an elected head of state.

Chronometrical Variables

Alias: *Duration*

Family: Continuous

Example: 2:56:15

The chronical specialization of an angular variable is called **chronometrical**. For example, the length of a country's anthem measured in minutes and seconds.

Independent *vs.* Dependent

In mathematical modeling, statistical modeling and experimental sciences, the values of *dependent variables* depend on the values of *independent variables*. The dependent variables represent the output or outcome whose variation is being studied. The independent variables represent inputs or causes, *i.e.*, potential reasons for variation or, in the experimental setting, the variable controlled by the experimenter. Models and experiments test or determine the effects that the independent variables have on the dependent variables. [Source: Wikipedia]

Some pairs of synonyms might help the reader distinguish between the two:

Independent variable	**Dependent variable**
Predictor variable	*Predicted variable*
Controlled variable	*Measured variable*
Explanatory variable	*Explained variable*
Exposure variable	*Outcome variable*
Input variable	*Output variable*
Regressor	*Regressand*
Dimension	*Measure*

Since the values of independent variables define the sets of experiments or measurements being conducted, independent variables must always be defined using discrete typologies. Consequently, whenever continuous variables are used as independent variables for experimentation protocols, they must be discretized first (*Cf.* Variable Discretization).

In contrast, dependent variables can be either discrete or continuous.

One way to better understand the fundamental dichotomy between independent and dependent variables is to consider the way group-by aggregations (*i.e.* pivots) are computed. Group-by aggregations are defined with two sets of variables:

- Group-by dimensions (*e.g. continent*)
- Pairs of measures and aggregation functions (*e.g. country population* and *sum*)

Clearly, the dimension against which country populations is being summed (*continent*) must be discrete, and the dataset defined by this group-by aggregation is defined as follows:

- One variable per group-by dimension (independent variable, always discrete)
- One variable per measure (dependent variable, discrete or continuous)

Typology Aggregation

When conducting a group-by aggregation on a measure, the typology of the aggregated measure is conditioned by the typology of the measure being aggregated and the aggregation function being used, as defined by the following table:

Aggregation Function	**Aggregation Typology**
Average	*Self*
Count	*Cardinal*
Max	*Self*
Min	*Self*
Quantile	*Quantile*
Standard Deviation	*Intensive*
Sum	*Self*
Sum of Squares	*Self* (with unit squared)
Variance	*Intensive*

Variable Discretization

In order to allow any independent variable to be used as dimension for a group-by aggregation whether it is defined with a discrete or continuous typology, *Principia Data* defines all continuous typologies with corresponding discretized typologies.

Continuous Typology	**Discretized Typology**
Intensive	*Incremental*
Extensive	*Cardinal*
Rational	*Quantile*
Longitudinal	*Meridional*
Percent	*Fractional*
Angular	*Sectorial*
Temporal	*Calendar*
Chronological	*Horological*
Chronometrical	*Horometrical*

The methods by which continuous variables can be discretized or discrete variables can be re-discretized depend on the source and target typologies, as well as the context within which such a discretization is conducted. Potentially, there is an infinite number of such discretization methods, and their study is beyond the scope of this paper. For more information, please refer to additional reference materials, including the documentation of products using *Principia Data* as foundational typology.

Precision vs. Parsimony

As suggested earlier (*Cf.* Terminal *vs.* Calendar), the selection of a typology for a given variable is driven by two contradictory principles: the *principle of precision*, which dictates that variables should be defined using typologies that are as precise as possible, and the *principle of parsimony*, which dictates that variables should be encoded in the simplest possible way in accordance to the objectives of the analysis being conducted.

Principle of Precision

For any given variable, the most precise typology suitable for the variable should be selected. This will ensure that a maximum number of mathematical operators are available to perform analysis on the variable, and a maximum number of visuals (*i.e.* charts) are available to produce visualizations for the variable. It will also ensure that proper discretization or re-discretization methods are available to make independent variables suitable as dimensions of group-by aggregations (*Cf.* Variable Discretization).

Principle of Parsimony

For any given variable, the simplest encoding should be used in order to simplify the variable's analysis by focusing on the most relevant subset of its properties (*Cf.* Occam's razor). Consequently, this principle of parsimony encourages variables to be defined with the least precise typology capable of reflecting this subset of properties. In most cases, raw variables will be encoded in source datasets using the most precise encodings so as not to lose any information (*date* instead of *term* for example). Nevertheless, data analysts should be encouraged to push variable encoding down through some graceful typology degradation path in order to keep data models as simple as possible, but no simpler.

Balancing Precision and Parsimony

Obviously, the principle of precision is in direct contradiction with the principle of parsimony. The former advocates for more precise typologies, while the latter advocates for lesser ones. This tension is a powerful force to help refine generic data models while simplifying them for the purpose of specific analysis scenarios. Balancing the two principles is the responsibility of the data analyst, and the following guidelines can be used to inform difficult decisions:

> Select the *boolean* typology whenever a *categorical* variable has a binary cardinality.
> (*Cf.* Boolean Variables)
>
> Select the *identificational* typology whenever a variable could be used as unique key.
> (*Cf.* Identificational Variables)
>
> Cardinality usually dictates the choice between *categorical* and *relational*.
> (*Cf.* Relational Variables)
>
> Prefer ordinal over sequential whenever possible (sequential is rarely applicable).
> (*Cf.* Sequential Variables)
>
> Select the *terminal* typology whenever the absolute epochal date does not matter.
> (*Cf.* Terminal vs. Calendar)
>
> Prefer *discrete* typologies to *continuous* ones to simplify future discretizations.
> (*Cf.* Variable Discretization)

Principia Data Framework

The data typology outlined so far is only one layer of the framework proposed by *Principia Data*. Above and below this layer are two additional layers, which together with typologies form the core *Principia Data Framework:*

Layer 3	*Datatypes*
Layer 2	*Typologies*
Layer 1	*Primitives*

Primitives

The *typologies* defined by *Principia Data* are designed to be technology agnostic. Nevertheless, they have been conceived with the assumption that values of most variables that can be defined against these typologies could be encoded within most computer systems and applications using four simple *primitives*:

- *Integer*
- *Float*
- *String*
- *Blob*

Different computer systems might use more or less primitives, but these four constitute a solid baseline that can be used for making a wide range of extremely useful simplifying assumptions. The default set of rules dictating the way typologies can be bound to primitives are outlined in the appendix (*Cf.* List of Typologies), even though such rules can vary across systems.

Datatypes

On top of the generic *typologies* defined by *Principia Data*, data analysts are free to develop specific datatypes defined using generic *typologies* and specific *options*. This is especially useful when dealing with variables that have contextually or culturally dependent properties such as email address (a specific type of *identificational* variable), date of birth (a calendar variable which values cannot be in the future), or street address (a variable that is best represented through the composition of several variables like *street number*, *street name*, *city*, *postal code*, and *country*).

For this purpose, Principia Picture defines three classes of datatypes:

- *Scalar Datatypes* for variables made of single values (*i.e.* scalars);
- *Vectorial Datatypes* for variables made of multiple values of the same typology;
- *Composite Datatypes* for variables made of multiple values of different typologies.

Scalar Datatypes

Scalar datatypes can be used to define variables which values are single scalars defined using the same typology. For example color, email address, hyperlink, icon, image, or password.

Scalar datatypes are usually defined with a set of options. For example, a color might be defined in relation to a specific color palette, or in accordance to a particular color model (*e.g.* RGB).

Vectorial Datatypes

Vectorial datatypes can be used to define variables which values are vectors of scalars defined using the same typology or datatype. Variables defined with a vectorial datatype can have a fixed size (for example the *x*, *y*, and *z* coordinates of a point in a 3D cartesian system), or a dynamic size (for example the list of signatories to international treaties).

The scalar values of vectorial datatypes can be defined using basic typologies, scalar datatypes, other vectorial datatypes (*e.g.* vector of vectors), or composite datatypes (*e.g.* polygon).

Composite Datatypes

Composite datatypes can be used to define variables which values are fixed-size vectors of scalars defined using several typologies or datatypes. For example, a geopoint can be defined as a pair of latitude and longitude, which are both defined using the longitudinal typology, but with different options (latitudes go from -90° to +90°, while longitudes go from -180° to +180°). Another example of composite datatype is the Google Maps geocoding of an address:

Component	**Typology or Datatype**
Street Number	*Sequential*
Route	*Lexical*
Locality	*Lexical*, *Denominational*, or *Relational*
Administrative Area Level 2	*Lexical*, *Denominational*, or *Relational*
Administrative Area Level 1	*Lexical*, *Denominational*, or *Relational*
Country	*Lexical*, *Denominational*, or *Relational*
Postal Code	*Identificational*
Formatted Address	*Textual*
Location	*Geopoint*
Place ID	*Identificational*

Interestingly, the composite datatype used to define an address includes a nested composite datatype (the geopoint used for the location). This is an illustration of the hierarchical nature of composite datatypes, and an indication of how powerful they can be to describe even the most complex data structures.

Another very common composite datatype is a *currency amount*, which is made of a currency (*identificational* or *relational*) and an amount (*intensive* or *extensive*).

Geospatial Datatypes

Using vectorial and composite datatypes, *Principia Data* can be used to describe most common geospatial datatypes. For example:

Datatype	**Definition**
Point	Vector of 2 *longitudinal* scalars
MultiPoint	Vector of *n points*
LineString	Vector of *n points*
MultiLineString	Vector of *n linestrings*
Polygon	Vector of *n points*
MultiPolygon	Vector of *n polygons*

Units of Measurement

Most variables defined with the *intensive* and *extensive* typologies are measured using standard or custom units of measurement. Fortunately, the International System of Units classifies units of measurement across intensive units and extensive units, which makes *Principia Data* suitable for the typing of continuous variables related to physical measures.

Tables and Views

To a large extent, *Principia Data* is technology agnostic, in the sense that it could be used as underlying data typology for virtually any database management system. Nevertheless, some typologies like *relational*, *hierarchical*, or *serial* imply some relational model, at least from a typological standpoint. Furthermore, higher-level grammars or frameworks built on top of *Principia Data* (Cf. Principia Pictura) might take advantage of the **table** and **view** concepts offered by the relational model. In such a context, *Principia Data* makes use of these concepts, assuming that they can be properly supported by any modern database management system, not just relational ones, with or without support for the SQL query language.

Tables

A **table** is a set of data elements using a model of vertical **columns** (identifiable by **name**) and horizontal **rows**, the cell being the unit where a row and column intersect. A table has a specified number of columns, but can have any number of rows. Each row is identified by one or more values appearing in a particular columns subset. The columns subset which uniquely identifies a row is called the **primary key**. [Source: Wikipedia]

According to this definition, there is a bijective relationship between *variables* and *columns*. Every column is defined with a *name*, a *primitive*, a *typology* or a *datatype*, and some *options*. Together, the column definitions for an entire table make a **table schema**.

The schema of a table is also referred to as **metadata** for the table.

Views

In database theory, a **view** is the result set of a stored query on the data. This pre-established query command is kept in the database dictionary. Unlike ordinary base tables in a relational database, a view does not form part of the physical schema: as a result set, it is a virtual table computed or collated dynamically from data in the database when access to that view is requested. Changes applied to the data in a relevant underlying table are reflected in the data shown in subsequent invocations of the view. [Source: Wikipedia]

Principia Data adopts a very similar model, but decomposes the *query* into four successive steps (*Cf.* View Structure) in order to make the creation of views even more declarative than it is enabled by the SQL query language.

Views can be described as virtual tables. As such, they are defined with a **view schema**, which has the exact same structure as a table schema. The main difference between a table and a view is a purely technical one: tables materialize their data (on some storage media or in memory), while views do not, unless they are explicitly materialized, in which case they become real tables (*Cf.* View Source). This fundamental distinction between tables and views with respect to the materialization of data has direct implications on the *CREATE*, *READ*, *UPDATE*, and *DELETE* operations that can be performed on table rows (*Cf.* CRUD Operations).

The schema of a view is also referred to as **metadata** for the view.

Column Encoding

The cell values of table columns are encoded on some storage media or in memory according to the *primitives* (*Cf.* Primitives) of their columns, as defined in the *table schema*. The cell values of view columns are encoded in a similar fashion in memory whenever a view is computed.

Virtual columns can be defined in order to support vectorial datatypes or composite datatypes (*Cf.* Vectorial Datatypes and Composite Datatypes), but the definition of their encoding is beyond the scope of this paper.

View Structure

With database query languages like SQL, a view is created using a single query. In contrast, Principia Data decomposes a view into four successive steps:

1. **Source** — to select the source table or source view for the view and add rows to it;
2. **Brew** — to add, modify, or remove columns in the view;
3. **Filter** — to filter out rows from the view;
4. **Pour** — to apply a transformation to the view such as *Identity* or *Pivot*.

This beverage-inspired metaphor (think beer or tea) implies a mandatory sequencing of steps, which is somehow arbitrary, yet motivated by a host of practical or technical reasons, which are outlined in the following chapters.

View Source

The **source** step of a view defines where its data comes from, out of three possible options:

- Another view
- A set of files
- A set of *ah hoc* rows

The first two options are mutually exclusive, but the third one can be used in a standalone manner, or complement either one of the first two. When a set of files is used as source for the view, a new table is created, and this table plays the role of **source table** for the view. The same is true when a set of *ad hoc* rows is used as source for the view without any other view or any files. In contrast, when an existing view is used as source for the new view, the *source table* for the new view is the one associated to the existing view, either directly or recursively through the parental relationship established between parent and child views.

When a first view is defined as source for a second view, the output of the first view's *pour* step is used as source for the second view by default. Nevertheless, the output of any step of the first view can be used as source for the second view as well, including its *source*, *brew*, or *filter* steps. This finer level of granularity in the sourcing of views is useful in certain instances and simplifies the end-to-end view hierarchy by removing some intermediary views that would be necessary if sources could be defined only from the output of the *pour* step.

The addition of *ad hoc* rows to the source of a view is always applied to the view's source table.

A view's schema is defined from the *source* step, then updated through the *brew* and *pour* steps.

View Brew

The **brew** step of a view is used to add, modify, or remove columns in the view. The brew step is executed after the source step, because the source step is responsible for initializing the metadata of the view and for populating its data. Without these, a brew step would have no existing columns upon which new columns could be calculated.

View Filter

The **filter** of a view is used to filter out rows from the view according to filtering facets defined in relation to individual columns. The filter step is executed after the brew step so that filtering facets can be applied to columns that were added or modified through the preceding brew step.

While *Principia Data* does not mandate a particular model for the definition and execution of filtering facets, the following model offers a simple foundation to build upon:

- A filter step is defined by one or multiple filtering facet;
- A filtering facet is defined in relation to one and only one column;
- No more than one filtering facet can be defined for any given column;
- Filtering facet are combined within a filter step through the boolean *AND* operation.

View Pour

The **pour** step of a view is used to apply a transformation to the view such as *Identity* or *Pivot*. The pour step is executed as very last step because it has the ability of transforming the schema of the view in some fundamental ways. While an infinite number of transformations could be envisioned, the following four offer a solid foundation to build upon:

- *Identity* — no transformation;
- *Materialize* — produce a new table from the filtered output of a view;
- *Pivot* — perform a group-by aggregation;
- *Custom* — perform an arbitrary aggregation defined with custom software code.

Some transformations such as *Materialize* or *Pivot* force the materialization of the view's output. In such a case, a new *source table* is created. It is therefore important to notice that source tables can be created either from the *source* step or from the *pour* step.

Some transformations like *Pivot* also produce a brand new schema for the view:

- One column per pivot dimension (independent variable, always discrete)
- One column per pivot measure (dependent variable, discrete or continuous)

(*Cf.* Independent vs. Dependent and Typology Aggregation)

CRUD Operations

The table model allows table rows to be created, read, updated, and deleted. These conventional CRUD operations can also be performed through a view, but when doing so, they are directly applied to the view's source table (*Cf.* View Source). The following examples illustrate them.

Example 1: *View 1* is defined with a materializing pour transformation (*e.g. Materialize* or *Pivot*). *View 1/1* and *View 1/2* use the output of View #1 as common source. If rows are added to *View 1/1,* they are actually added to the table materialized by the *pour* step of *View 1*. This table is considered as source table for both *View 1/1* and *View 1/2*. As such, rows added to *View 1/1* are also visible through *View 1/2*. In fact, rows are not really added *to View 1/1,* but rather added *through* it.

Example 2: *View 1* is defined with a *Pivot* transformation. One of the dimensions *Δ* of this pivot has three values, *a*, *b*, and *c*. *View 1/1* uses the output of *View 1* as source and adds a set of rows to the table materialized by the *pour* step of *View 1*. This new set of rows is defined in such a way that it extends the pivot's output as if *Δ* had a fourth value *d*. From there, another set of rows is added from the source step of *View 1*, in such a way that it extends *Δ* with a fourth value *d*. As a result, the pivot results returned through *View 1/1* will have duplicate entries in regards to value *d* of dimension *Δ*. This is to be expected, and it is not a symptom of a deficient architecture. Instead, it is a cautionary warning that adding *ad hoc* rows to pivot results is questionable.

Example 3: *View 1* is defined with a *Pivot* transformation. One of the dimensions *Δ* of this pivot has three values, *a*, *b*, and *c*. *View 1/1* uses the output of *View 1* as source and updates all resulting pivot measures related to value *a* of dimension *Δ*. From there, the rows of *View 1* related to value *a* of dimension *Δ* are updated as well, but in ways that are not consistent with the corresponding updates made from *View 1/1*. Because updates made through *View 1/1* are downstream of the updates made through *View 1*, the former prevail over the latter.

Principia Pictura

One of the main purposes of carefully typing statistical variables is the facilitation of their visualization through charts, maps, and other visuals. In such a context, the *Principia Data* unified typology of statistical variables serves as typological foundation for the *Principia Pictura* unified grammar of charts. *Principia Pictura* is a framework for:

- Providing a formal definition of charts, maps, and other visuals;
- Enumerating and defining all basic charts;
- Defining visual transformations for producing advanced charts;
- Binding the variables of datasets onto the axes of charts;
- Defining compatibility rules between variable typologies and axis properties;
- Recommending the most appropriate chart for a given dataset;
- Producing conventional statistical plots with standard charts;
- Suggesting common user interactions for producing charts with software tools.

List of Typologies

Family	Typology	Alias	Class	Primitive	Example
Discrete	*Boolean*	Boolean	Disc.	Integer	TRUE, FALSE
	Nominal	String	Disc.	String	Foo, Bar, Baz
	Textual	Text	Disc.	String	Lorem ipsum dolor sit amet
	Denominational	Name	Disc.	String	John Doe
	Identificational	Identifier	Disc.	String	stoic.com
	Categorical	Category	Disc.	Integer	Red, Green, Blue
	Relational	Relation	Disc.	Integer	USA$^{\times}$, Mexico$^{\times}$, Canada$^{\times}$
	Hierarchical	Hierarchy	Disc.	Integer	World$^{\times}$, North America$^{\times}$, USA$^{\times}$
	Serial	Series	Disc.	Integer	George Washington$^{\times}$, John Adams$^{\times}$
	Lexical	Word	Disc.	String	Alpha, Bravo, Charlie
	Sequential	Sequence	Disc.	Integer	1st, 2nd, 3rd
	Ordinal	Index	Disc.	Integer	1, 2, 3
	Incremental	Increment	Disc.	Integer	10, 20, 30
	Cardinal	Count	Disc.	Integer	10, 100, 1000
Continuous	*Intensive*	Level	Cont.	Float	10°C, 20°C, 30°C
	Extensive	Amount	Cont.	Float	10m^2, 100m^2, 1000m^2
Absolute	*Quantile*	Quantile	Disc.	Integer	1st Quartile, 2nd Quartile
	Rational	Ratio	Cont.	Float	0.25, 0.5, 0.75
	Fractional	Fraction	Disc.	Integer	¼, ½, ¾
	Percent	Percent	Cont.	Float	25%, 50%, 75%
Directional	*Meridional*	Meridian	Disc.	Integer	0°, 45° W, 90°E
	Longitudinal	Longitude	Cont.	Float	2° 20' 14.025" E
	Sectorial	Sector	Disc.	Integer	10°, 20°, 30°
	Angular	Angle	Cont.	Float	57.2958°
Epochal	*Terminal*	Term	Disc.	Integer	1st Term, 2nd Term, 3rd Term
	Calendar	Date	Disc.	Integer	January 1, 1970
	Temporal	Timestamp	Cont.	Float	January 1, 1970, 00:00:00
Chronical	*Horological*	Moment	Disc.	Integer	2:56 AM
	Chronological	Time	Cont.	Float	2:56:15 AM
	Horometrical	Extent	Disc.	Integer	2:56
	Chronometrical	Duration	Cont.	Float	2:56:15