STAT-341
Computational Statistics and Data Analysis

David Duan

University of Waterloo

Contents

1 Populations

Our goal is to describe a population using attributes.

- A **population** is a finite (though possibly huge) set $\mathcal{P}$ of elements.
- Elements of a population are called **units**, denoted $u \in \mathcal{P}$.
- **Variates** are functions $x(u)$, $y(u)$, etc., on the individual units $u \in \mathcal{P}$. We often use the notation x_u, y_u, etc., when referring to the *realized values* of these variates for unit $u \in \{1, ..., N\}$.

In the next section, we will define and explore **population attributes**, denoted generally as $a(\mathcal{P})$ (some text uses α instead of a). In particular, we will define them, consider how to calculate them, and evaluate some of their (non-sampling) properties, e.g., interpreting the characteristic being captured, sensitivity to outlying points, etc.

2 Explicit Attributes

2.1 Population Attributes

We formally define the following terms seen in the previous section:

- A **population** is a set or collection of **units**, with one or more *variates* that we can measure.
- **Variates** are characteristics (can be *numerical* or *categorical*) of each unit in the population.
- **Population attributes** are summaries that describe the characteristics of the population. More formally, an attribute is a function that is applied to the whole population and determined by the variate values observed for each of the population's units.

Suppose we are interested in the variate ys of a population $\mathcal{P} = \{u_1, \ldots, u_N\}$. Then the actual variate values realized by the units are $\{y_1, \ldots, y_N\}$, and we can denote a population attribute derived from a function f as

$$\boxed{a(P) = f(y_1, \ldots, y_N).}$$

For example, we can compute the *population total* as:

$$a(P) = \sum_{u \in P} y_u.$$

We can also calculate the number of units that satisfy a given predicate I_A as:

$$a(P) = \sum_{u \in P} I_A(y_u).$$

In general, attributes can be *numerical* or *graphical*, as long as they summarize the whole population. Below are all attributes of a population:

- A histogram of y_u values.
- A Scatter-plot of the (x_u, y_u) pairs.
- The least square estimate of the line-of-best-fit.
- The residual variation around the line-of-best-fit.

▷ Location Attributes

These attributes measure or describe the *center* of the distribution of variate values in a given dataset.

- the **population average**:

$$\boxed{a(\mathcal{P}) = \hat{y} = \frac{1}{N} \sum_{u \in \mathcal{P}} y_u.}$$

- the **population proportion** (with respect to some predicate I_A):

$$\boxed{a(\mathcal{P}) = \frac{1}{N} \sum_{u \in \mathcal{P}} I_A(y_u).}$$

Other examples include the *population mode*, the *population median*, etc.

▷ Spread Attributes

These attributes measure *variability* or *spread* of variate values in a given dataset.

- the **population variance**:

$$\boxed{a(\mathcal{P}) = \frac{1}{N}\sum_{u\in\mathcal{P}}(y_u - \overline{y})^2.}$$

- the **population standard deviation**:

$$\boxed{a(\mathcal{P}) = SD_{\mathcal{P}}(u) = \sqrt{\frac{\sum_{u\in\mathcal{P}}(y_u - \overline{y})^2}{N}}.}$$

- **coefficient of variation**:

$$\boxed{a(\mathcal{P}) = \frac{SD_{\mathcal{P}}(y)}{\overline{y}}.}$$

Remark. The variance and standard deviation can also be defined using $N-1$ as denominator. We would further discuss about the comparison between N and $N-1$.

▷ Order Statistics

Population attributes can also be based on an indexed collection of values,

$$\boxed{y_{(1)} \le y_{(2)} \le \dots \le y_{(N)}}$$

which are the variate values in population that are ordered from the smallest to largest (including ties.) The set $\{y_{(i)}\}_{i=1}^N$ can be viewed as a permutation of $\{y_i\}_{i=1}^N$, sorted based on some criteria.

▷ Location Attributes based on Order Statistics

- the **population minimum**:

$$\boxed{a(\mathcal{P}) = \min_{u\in\mathcal{P}} y_u = y_{(1)}.}$$

- the **population maximum**:

$$\boxed{a(\mathcal{P}) = \max_{u\in\mathcal{P}} y_u = y_{(N)}.}$$

- the **population mid-range**:

$$\boxed{a(\mathcal{P}) = \frac{1}{2}[y_{(1)} + y_{(N)}].}$$

- the **population median**:

$$\boxed{a(\mathcal{P}) = \text{median}_{u\in\mathcal{P}}\;\; y_u.}$$

- the **population quartiles**:
  - Q_1 is 25^{th} percentile, or the first quartile;
  - Q_2 is 50^{th} percentile, or the median;
  - Q_3 is 75^{th} percentile, or the third quartile.

▷ Variability Attributes base on Order Statistics

- the **population range**:

$$\boxed{a(\mathcal{P}) = \max_{u\in\mathcal{P}} y_u - \min_{u\in\mathcal{P}} y_u = y_{(N)} - y_{(1)}.}$$

- the **population inter-quartile range** (IQR):

$$\boxed{a(\mathcal{P}) = Q_3 - Q_1.}$$

- the **median absolute deviation** (MAD):

$$\boxed{a(\mathcal{P}) = \text{median}_{u\in\mathcal{P}} \left| y_u - \text{median}_{u\in\mathcal{P}}\ y_u \right|;}$$

  that is, MAD is the median of the absolute differences between each y_u and the population median.

- the **average squared distance from the mean** is

$$\boxed{a(\mathcal{P}) = \frac{1}{N}\sum_{u\in\mathcal{P}} (y_u - \overline{y})^2.}$$

▷ Skewness Attributes

These are measures of *asymmetry* in a population. A symmetric distribution of the population values should result in a skewness attributes of zero.

- **Pearson's moment coefficient of skewness**:

$$\boxed{a(\mathcal{P}) = \frac{\frac{1}{N}\sum_{u\in\mathcal{P}}(y_u - \overline{y})^3}{[SD_{\mathcal{P}}(y)]^3}.}$$

  This metric focus on the average and is *dimensionless*.

- **Pearson's second skewness coefficient**, also known as the **median skewness**:

$$\boxed{a(\mathcal{P}) = 3 \times \frac{(\bar{y} - \text{median}_{u\in\mathcal{P}}\, y_u)}{SD_{\mathcal{P}}(y)}.}$$

- **Bowley's measure of skewness** (based on the quartiles):

$$\boxed{a(\mathcal{P}) = \frac{(Q_3 + Q_1)/2 - Q_2}{(Q_3 - Q_1)/2}.}$$

▷ NAs in R

Many programs in R accommodate missing data (represented as `NA`s) and do something appropriate (typically omit them).

- For your own code and analysis, you either need to decide what to do with `NA`s or ensure that the data do not have any `NA`s.
- If you choose to omit `NA`s for example using the function `na.omit(...)` may be helpful. It will remove rows which contain any `NA` from a data set.
- For other possibilities, please see `help("na.omit")` or `?na.omit` in R.

2.2 Attribute Properties

Let us examine the behaviour of an attribute when we change the units and duplicate the population. Recall that a *population attribute* is a function of measured variates y_u:

$$a(\mathcal{P}) = f(y_1, ..., y_N),$$

and the variates y_u are typically associated with some measurement units. In general, we are interested in how an attribute changes when we change the units of measurement and when we change the population.

There are four categories of changes:

- Change in **scale** of measurement, e.g., meter to millimeter.
- Change in **location** of the zero of measurement, e.g., 0°K vs −273°C.
- Change in *both* scale and location, e.g., 0°K → −273°C and 1°C = 1.8°F.
- Change in *more than* just scale and location: miles per hour vs kilometers per minutes.

▷ Location Invariance and Equivariance

For an attribute $a(\mathcal{P}) = a(y_1, ..., y_N)$, we say that for any $m > 0$ and $n \in \mathbb{R}$, that the attribute is

- **location invariant** if
$$\boxed{a(y_1 + b, ..., y_N + b) = a(y_1, ..., y_N).}$$
- **location equivariant** if
$$\boxed{a(y_1 + b, ..., y_N + b) = a(y_1, ..., y_N) + b.}$$

Example. The population average is location equivariant, i.e., increasing all variate values by a constant b would lead to an increment of b in the population average.

▷ Scale Invariance and Equivariance

For an attribute $a(\mathcal{P}) = a(y_1, ..., y_N)$, we say that for any $m > 0$ and $b \in \mathbb{R}$, the attribute is

- **scale invariant** if
$$\boxed{a(my_1, ..., my_N) = a(y_1, ..., y_N) = a(\mathcal{P}).}$$
- **scale equivariant** if
$$\boxed{a(my_1, ..., my_N) = ma(y_1, ..., y_N) = ma(\mathcal{P}).}$$
- **Location-scale invariant** if it is both location invariant and scale invariant, i.e.
$$\boxed{a(y_1 m + b, ..., my_N + b) = a(y_1, ..., y_N) = a(\mathcal{P}).}$$
- **location-scale equivariant** if it is both location equivalent and scale equivariant, i.e.
$$\boxed{a(y_1 m + b, ..., my_N + b) = ma(y_1, ..., y_N) + b = ma(\mathcal{P}) + b.}$$

Example. The population average is location-scale equivariant:

$$a(my_1 + b, ..., my_N + b) = \frac{1}{N}\sum_{u \in \mathcal{P}}(my_u + b) = m\left(\frac{1}{N}\sum_{u \in \mathcal{P}}\right) + b = ma(\mathcal{P}) + b.$$

▷ Replication

Another invariance/equivariance property of interest for population attributes is **replication invariance** and **replication equivariance**, i.e., if a population $\mathcal{P}$ is duplicated $k-1$ times (so that there are k copies of it), how does the attribute change on this new population, denoted by $\mathcal{P}^k$?

$$\begin{aligned}
\mathcal{P} &= \{y_1, ..., y_N\} \text{ with size } N \\
\mathcal{P}^2 &= \{y_1, ..., y_N, y_1, ..., y_N\} \text{ with size } 2N \\
&\cdots \\
\mathcal{P}^k &= \{y_1, ..., y_N, y_1, ..., y_N, y_1, ..., y_N\} \text{ with size } kN
\end{aligned}$$

We say the attribute $a(\mathcal{P})$ is:

- **replication invariant** whenever $a(\mathcal{P}^k) = a(\mathcal{P})$.
- **replication equivariant** whenever $a(\mathcal{P}^k) = k \cdot a(\mathcal{P})$.

For example, the population average is replication invariant:

$$\begin{aligned}
\mathcal{P}^k &= \{x_1, ..., x_{kN}\} \\
a(\mathcal{P}^k) &= \frac{1}{kN} \sum_{j=1}^{kN} x_j \\
&= \frac{1}{kN} \sum_{u \in \mathcal{P}} k \cdot y_u \\
&= \frac{1}{N} \sum_{u \in \mathcal{P}} y_u \\
&= a(\mathcal{P}).
\end{aligned}$$

2.3 Influence, Sensitive Curves, and Breakdown Points

One way to study a population attribute $a(\mathcal{P}) = a(y_1, ..., y_N)$, is to consider the effect of *adding* or *removing* a single variate y_u to examine its impact on $\mathcal{P}$. To quantify this effect, we look at the difference in the attribute when the variate value is added (*sensitivity*) or removed (*influence*). We also consider a special form of sensitivity/influence measure called the *breakdown point*.

▷ Influence

The **influence** of a variate y_u (corresponding to unit $u \in \mathcal{P}$) on the population attribute $a(\mathcal{P})$ is quantified by the difference between the attribute value when y_u is included and the attribute value when y_u is removed:

$$\Delta(a, u) = a(y_1, ..., y_{u-1}, y_u, y_{u+1}, ..., y_N) - a(y_1, ..., y_{u-1}, y_{u+1}, ..., y_N).$$

Ideally, no single unit's value should be greater influence than any other. If a unit had larger influence than the rest, it would require further investigation because it could either be an error or it's the most interesting unit in the population.

Example. Consider the influence of a variate y_u on the population average. We start by deriving the average without unit u:

$$a(y_1, ..., y_{u-1}, y_{u+1}, ..., y_N) = \frac{1}{N-1} \sum_{k \in \mathcal{P}, k \neq u} y_k = \frac{\sum_{k \in \mathcal{P}} y_k - y_u}{N-1} = \frac{N\overline{y} - y_u}{N-1}.$$

The influence for a given u is thus

$$\Delta(a, u) = \overline{y} - \frac{N\overline{y} - y_u}{N-1} = \frac{(N-1)\overline{y} - N\overline{y} + y_u}{N-1} = \frac{y_u - \overline{y}}{N-1}.$$

To calculate this in R, we can use a loop:

```
delta = rep(0, length(y))
for (i in 1:length(y)) {
    delta[i] = ybar - mean(y[-i])
}
```

- The `rep(x, k)` replicates value x for k times. In this case, it builds a zero vector of size `length(y)`.
- `y[-i]` removes the ith element from a vector.

▷ Sensitivity

We can examine the effect on a population attribute when we add a variate. To examine this effect, suppose we have a population of size $N-1$, $\mathcal{P} = \{y_1, ..., y_{N-1}\}$, and add a variate with value y. Then our new population with N elements is $\mathcal{P}^* = \{y_1, \ldots, y_{N-1}, y\}$. Define the **sensitivity curve** of an attribute as

$$\boxed{SC(y; a(\mathcal{P})) = \frac{a(\mathcal{P}^*) - a(\mathcal{P})}{1/N} = N(a(\mathcal{P}^*) - a(\mathcal{P}))}$$

We can then plot the sensitive curve as a function of the new variate value y, which gives a scaled measure of the effect that a single variate value y has on the value of a population attribute $a(\mathcal{P})$. The sensitivity curve for any attribute can be determined mathematically in general, but can also be determined computationally for any particular population and any particular attribute.

The following function computes the sensitivity curve for any population and any attribute in R:

```
sc = function(y.pop, y, attr,...) {
    N <- length(y.pop) + 1
    sapply(y, function(y.new)) {
        N * (attr(c(y.new, y.pop), ...) - attr(y.pop, ...))
    }
}
```

- `y.pop`: A vector of length $N-1$, which contain variate values of population units.
- `y`: The new variate value (corresponding to the new unit).
- `attr`: The function f that maps variate values to a population attribute value, i.e., $a(\mathcal{P}) = \texttt{attr}(\ldots)$.
- `sapply`: Given a list, vector, or data frame as input, return output in vector or matrix. It is useful for operations on list objects and returns a list object of same length of original set.

Example. Consider population mean.

$$a(\mathcal{P}) = \frac{1}{N-1}\sum_{i=1}^{N-1} y_i = \bar{y}_{N-1}$$
$$a\left(\mathcal{P}^*\right) = \frac{1}{N}\left(\sum_{i=1}^{N-1} y_i + y\right) = \frac{(N-1)\bar{y}_{N-1} + y}{N}$$

The sensitive curve is given by

$$\begin{aligned} SC(y) &= N\left[a\left(\mathcal{P}^*\right) - a(\mathcal{P})\right] \\ &= N\left(\frac{N-1}{N}\bar{y}_{N-1} + \frac{y}{N} - \bar{y}_{N-1}\right) \\ &= (N-1)\bar{y}_{N-1} + y - N\bar{y}_{N-1} \\ &= y - \bar{y}_{N-1} \end{aligned}$$

The sensitivity curve gets higher (or lower) without bound as $y \to \infty$ (or as $y \to -\infty$). Thus, a single observation can change the average by a huge (or even infinite) amount.

Example. Consider population maximum.

$$a(P) = \max\{y, \ldots, y_{N-1}\} = y_{(N-1)}$$
$$a\left(P^*\right) = \max\{y,, \cdots, y_{N-1}, y\} = \begin{cases} y_{(N-1)} & \text{if } y < y_{(N-1)} \\ y & \text{if } y > y_{(N-1)} \end{cases}$$

Then

$$SC(y) = N(a(\mathcal{P}^*) - a(\mathcal{P})) = \begin{cases} 0 & \text{if } y < y_{(N-1)} \\ N(y - y_{(N-1)}) & \text{if } y \geq y_{(N-1)} \end{cases}$$

Note that the curve is unbounded for large y, which means the maximum is sensitive to large values.

Example. Consider 2nd order statistic.

$$a(\mathcal{P}) = y_{(2)}$$
$$a(\mathcal{P}^*) = \begin{cases} y_{(1)} & y < y_{(1)} \\ y & y_{(1)} \leq y \leq y_{(2)} \\ y_{(2)} & y \geq y_{(2)} \end{cases}$$

Then

$$SC(y) = \begin{cases} N(y_{(1)} - y_{(2)}) & y < y_{(1)} \\ N(y - y_{(2)}) & y_{(1)} \leq y \leq y_{(2)} \\ 0 & y \geq y_{(2)} \end{cases}$$

Observe that this is a bounded function.

▷ Breakdown Points

Another measure of robustness is called the *breakdown point*, which gives an assessment of just how large a proportion of the data must be contaminated before the statistic breaks down.

The **breakdown point** of a statistic is the smallest possible fraction of the observations that can be changed to something very extreme (plus or minus infinity) to make the error large (infinite). For example,

- the break-point for the average is $1/N$ (or asymptotically zero), because even one element being infinity will break the average.
- the break-point for the median is $1/2$ (i.e., half of the data has to go to infinity before the median breaks down).

An attribute with high breakdown points are called **resistant** or **robust**.

2.4 Graphical Attributes

Population attributes can also be entirely graphical as in

- histograms of y_u values
- bar plots of y_u values
- box plots of y_u values
- scatter-plots of pairs (x_u, y_u)
- scatter-plots of quantiles and ranks of y_u (quantile-plots)

Each of them summaries the population, so they are all attributes.

▷ **Histograms**

Histograms help determine how the values are concentrated.

Consider the population $\mathcal{P} = \{y_1, \dots, y_N\}$. To plot a histogram, partition the range of the population into k non-overlapping intervals, called **bins**, $I_j = [a_{j-1}, a_j)$, $j = 1, \dots, k$ and then calculate the number or proportion of observations in the jth bin for $j = 1, \dots, k$.

We can define bins two ways:

- bins of equal size, or (this is common)
- bins with equal number of elements but varying size (less common but can be very informative).

A few remarks on the influence of bins:

- Notice also how varying the bin size changes the *coarseness* of the histogram.
- For the histograms that has varying size, the areas of all rectangles in each panel are the same.
- Bins with equal number of elements but varying size can help identify asymmetry in the population.

Rules for the number of bins (with equal width):

- **Sturges rule:** More bins are needed as the population increases.

$$\boxed{\text{Bin size} = \lceil \log_2(N) + 1 \rceil}$$

- **Freedman-Diaconis rule:**

$$\boxed{\text{Bin size } = 2\frac{IQR(x)}{N^{1/3}}}$$

- **Scott's rule:**

$$\boxed{\text{Bin size } = 3.5\frac{\sigma}{N^{1/3}}}$$

Plotting histograms using raw data gives more interpretable results, whereas plotting histograms using transformed data gives more symmetric result.

▷ Scatter-plots

A **scatter-plot** is a plot of the points (x_u, y_u) for all units in the population. It is used to see whether two variates x and y are related in some way. Sometimes, the scatter-plot of a transformed version of the data provides more insights. We will discuss this later.

A common problem with scatter-plots is that for integer-valued variates, duplicated values are difficult to identify. One solution is to change the shading and vary the size of bullets, so that (for example) a location with multiple points will be darker than other spots. Alternatively, we could add **jitter** to the population values, which separates duplicate points slightly (provided it makes sense to do so):

$$y_u^* = y_u + \text{noise}.$$

The amount of jitter (noise) can vary depending on the data set.

2.5 Power Transformations

Given a variate y, it is sometimes helpful to re-express the values in a non-linear way via a transformation $T(y)$ so that on the transformed scale attributes are easier to define, understand, and determine.

A commonly used transformation when $y > 0$ is the family of **power transformations**, indexed by a power α:

$$\boxed{T_\alpha(y) = \frac{y^\alpha - 1}{\alpha} = \begin{cases} y^\alpha & \alpha > 0 \\ \log(y) & \alpha = 0 \\ -(y^\alpha) & \alpha < 0 \end{cases}}$$

These transformations are monotonic, in the sense that

$$y_u < y_v \iff T_\alpha(y_u) < T_\alpha(y_v).$$

In other words, these transformations preserve the order of variate values associated with units u and v.

To perform the power transformation in R:

```
powerfun <- function(x, alpha) {
    if (sum(x <= 0) > 0) stop("x must be positive")
    if (alpha == 0) log(x)
    else if (alpha > 0) x^alpha
    else -x^alpha
}
```

The α values can take any real value in principle, but we restrict it to a small set to make the results more interpretable:[1]

$$\{\ldots, -2, -1, -1/2, -1/3, 0, 1/3, 1/2, 1, 2, \ldots\}.$$

How do we pick α? Two different but related effects of transformations are often of interest:

1. Produce a more symmetric-looking histogram.
2. Produce a roughly linear scattler-plot.

Each effect provides us a rule that indicates whether we should move left/right in the above set.

1. The "center" of the histogram tells you which way to move:
   - If the center is on the left, then move the power "left" in the set.
   - If the center is on the right, then move the power "right" in the set.
2. The "center" corresponds to the curvature appearing in the scatter-plot. Since we have two variables x and y, we will have two ladders.
   - If the center is in quadrant 1, then move up on X and up on Y.
   - If the center is in quadrant 2, then move down on X and up on Y.
   - If the center is in quadrant 3, then move down on X and down on Y.
   - If the center is in quadrant 4, then move up on X and down on Y.

[1] The instructor referred to this set as Tukey's ladder, as he suggested that we could imagine that the set of powers were arranged in a ladder with the smallest powers on the bottom and the largest on the top. Thus, moving "up" or "down" on the ladder of powers corresponding to moving "right" or "left" in the set below, respectively.

2.6 Order Rank Quantiles

Recall the order statistic:

$$y_{(1)} \leq y_{(2)} \leq \cdots \leq y_{(N)}$$

Define the **rank statistic** $r_1, \ldots, r_N$ to be the ranks of the variate values $y_1, \ldots, y_N$. In particular, if $y_i = y_{(k)}$, then y_i is the kth smallest value, so y_i has rank $r_i = k$. Equivalently,

$$\forall u \in \mathcal{P} : y_{(r_u)} = y_u.$$

▷ **Variate Value vs Rank**

It is often helpful to draw a scatter-plot for y_u (y-axis) vs rank (x-axis), i.e., (r_u, y_u) or $(u, y_{(u)})$.

- The height at any point tells the location of the value of y.
- The horizontal location identifies where in the order of the variate values that unit appears.
- The plot is monotonically non-decreasing from left to right.
- Flat spots indicate tied values of y; nearly flat spots are values close to each other; rapid rising spots are values that are far away from each other.

▷ **Quantiles**

Rather than using ranks, it can be more convenient to use the proportion of units in the population having a value less-than-or-equal-to y. Thus, instead of plotting the pairs $(x, y) = (r_u, y_u)$, we could equivalently plot the pairs $(x, y) = (p_u, y_u)$ where

$$p_u = \frac{r_u}{N}$$

denotes the proportion of the units in $\mathcal{P}$ whose values are less than y_u.

▷ **Quantile Function**

Strictly speaking, the plotted points are $(p, Q_y(p))$ where $p \in \{1/N, 2/N, \ldots, 1\}$ and $Q_y(p)$ is the pth *quantile* of y. The function

$$Q_y(p) = y_{(N \times p)}$$

is sometimes called the **quantile function** of y for all $p \in [1/N, 1]$. Note that the quantile $Q_y(p)$ for any p locates the variate values in the population and is thus a measure of location. Most (but not all) location measures try to capture *central tendency*.

▷ **Deriving Population Attributes from the Quantile Function**

The quantile function is a population attribute which can be used to generate a number of other interesting population attributes. Let's start with location attributes, i.e., quantiles that measure center:

- **Median**: $Q_y(1/2)$.
- **Mid-hinge** (average of Q_1 and Q_3):

$$\boxed{\frac{1}{2}(Q_y(1/4) + Q_y(3/4)).}$$

- **Mid-range** (average of min and max):

$$\boxed{\frac{1}{2}(Q_y(1/N) + Q_y(1)).}$$

- **Trimean**:

$$\boxed{\frac{1}{4}(Q_y(1/4) + 2 \cdot Q_y(1/2) + Q_y(3/4))}$$

These values can be readily obtained from the quantile plot. For example, you can find Q_1 by reading the y-value corresponding to $x = 0.25$.

Next, we have derive attributes that measure *spread*:

- Range: $Q_y(1) - Q_y(1/N)$.
- IQR: $Q_y(3/4) - Q_y(1/4)$.

Alternatively, the difference between any two quantiles might be divided by the difference in the corresponding p values, i.e., the slope of the line segment joining any two points $(p_1, Q_y(p_1))$ and $(p_2, Q_y(p_2))$ for $p_1 < p_2$ provides a measure of spread.

3 Implicit Attributes

So far, we have defined population attributes $a(\mathcal{P})$ to be a summary of the population $\mathcal{P}$. These attributes are said to be *explicit* because they are defined explicitly by the population variates. In this section, we will discuss *implicit* attributes, which also summarize the population $\mathcal{P}$ but are defined only implicitly by the population variates. In particular, such attributes are solutions to an optimization problem. In addition, we will look at an regression example which incorporates influence.

3.1 The Minimum of a Function

In most practical situations, we are interested in a (possibly vector-valued) attribute $\boldsymbol{\theta}$ which minimizes some function $\rho(\boldsymbol{\theta}; \mathcal{P})$ of the variates in the population. In other words, we want the value $\widehat{\boldsymbol{\theta}}$ which satisfies

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\arg\min}\, \rho(\boldsymbol{\theta}; \mathcal{P})$$

The most common form for $\rho(\boldsymbol{\theta}, \mathcal{P})$ is a sum of functions $\rho(\boldsymbol{\theta}, u)$ evaluated at each unit $u \in \mathcal{P}$:

$$\rho(\boldsymbol{\theta}, \mathcal{P}) = \sum_{u \in \mathcal{P}} \rho(\boldsymbol{\theta}, u).$$

Remark. We only need to consider minimization here because

$$\underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\arg\max} \rho(\boldsymbol{\theta}; \mathcal{P}) = \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\arg\min} - \rho(\boldsymbol{\theta}; \mathcal{P}).$$

▷ Scalar-Valued Attributes

Common examples for a scalar-valued attribute $\theta \in \mathbb{R}$ and $u \in \mathcal{P}$:

- Least-squares: For $\rho(\theta; u) = (y_u - \theta)^2$, we have:

$$\hat{\theta} = \underset{\theta \in \mathrm{R}}{\arg\min} \rho(\theta, \mathcal{P}) = \underset{\theta \in \mathrm{R}}{\arg\min} \sum_{u \in \mathcal{P}} \rho(\theta, u) = \underset{\theta \in \mathrm{R}}{\arg\min} \sum_{u \in \mathcal{P}} (y_u - \theta)^2 = \bar{y}.$$

- Weighted least-squares: For $\rho(\theta, u) = w_u(y_u - \theta)^2$, we have

$$\hat{\theta} = \underset{\theta \in \mathrm{R}}{\arg\min} \rho(\theta, \mathcal{P}) = \underset{\theta \in \mathrm{R}}{\arg\min} \sum_{u \in \mathcal{P}} \rho(\theta, u) = \underset{\theta \in \mathrm{R}}{\arg\min} \sum_{u \in \mathcal{P}} w_u (y_u - \theta)^2 = \frac{\sum_{u \in \mathcal{P}} w_u y_u}{\sum_{u \in \mathcal{P}} w_u}.$$

- Least absolute deviations: For $\rho(\theta, u) = |y_u - \theta|$, we have

$$\hat{\theta} = \underset{\theta \in \mathrm{R}}{\arg\min} \rho(\theta, \mathcal{P}) = \underset{\theta \in \mathrm{R}}{\arg\min} \sum_{u \in \mathcal{P}} \rho(\theta, u) = \underset{\theta \in \mathrm{R}}{\arg\min} \sum_{u \in \mathcal{P}} |y_u - \theta| = Q_y(1/2).$$

- Least generalized absolute deviations: Using the vee function for some $q \in (0, 1)$,

$$\rho_q(\theta; u) = \begin{cases} q\,(y_u - \theta) & \text{if } y_u \geq \theta \\ (q-1)\,(y_u - \theta) & \text{if } y_u < \theta \end{cases}$$

  we have

$$\hat{\theta} = \underset{\theta \in \mathrm{R}}{\arg\min} \rho(\theta, \mathcal{P}) = \underset{\theta \in \mathrm{R}}{\arg\min} \sum_{u \in \mathcal{P}} \rho(\theta, u) = \underset{\theta \in \mathrm{R}}{\arg\min} \sum_{u \in \mathcal{P}} \rho_q(\theta; u) = Q_y(q).$$

▷ Vector-Valued Attribute: Simple Linear Regression

Consider **simple linear regression**: [2]

$$y_u = \alpha + \beta(x_u - c) + r_u \qquad u \in \mathcal{P} = \{(x_1, y_1), \ldots, (x_N, y_N)\}.$$

The attribute of interest is $\boldsymbol{\theta} = (\alpha, \beta)$ and are determined implicitly by

$$\widehat{\boldsymbol{\theta}} = (\widehat{\alpha}, \widehat{\beta}) = \underset{(\alpha,\beta)\in\mathbb{R}^2}{\arg\min} \sum_{u\in\mathcal{P}} \left(y_u - \alpha - \beta\left(x_u - c\right)\right)^2$$

In STAT-331, we show that

$$\widehat{\alpha} = \bar{y} - \widehat{\beta}(\bar{x} - c) \quad \text{and} \quad \widehat{\beta} = \frac{\sum_{u\in\mathcal{P}} \left(x_u - \bar{x}\right)\left(y_u - \bar{y}\right)}{\sum_{u\in\mathcal{P}} \left(x_u - \bar{x}\right)^2}$$

which help us determine the least-squares fitted line:

$$y = \widehat{\alpha} + \widehat{\beta}(x - c).$$

The equation of fitted values, defined for all $u \in \mathcal{P}$, is $\hat{y}_u = \widehat{\alpha} + \widehat{\beta}\left(x_u - c\right)$, and the residuals are given by the signed difference of actual value and the prediction:

$$\hat{r}_u = y_u - \widehat{\alpha} - \widehat{\beta}\left(x_u - c\right).$$

▷ Vector-Valued Attribute: Weighted Least Squares

In **weighted least squares**, the fitted line minimizes the following objective function:

$$(\widehat{\alpha}, \widehat{\beta}) = \underset{(\alpha,\beta)\in\mathbb{R}^2}{\arg\min} \sum_{u\in\mathcal{P}} w_u \left[y_u - \alpha - \beta\left(x_u - c\right)\right]^2$$

It is assumed the weights are known but, as we will see, the residuals from an ordinary LS regression model can help us determine sensible values.

As in ordinary LS regression, we need to determine a value for c. In this setting, it is common to either set $c = 0$ or define it to be the weighted average of x_u:

$$c = \bar{x}_w = \frac{\sum_{u\in\mathcal{P}} w_u x_u}{\sum_{u\in\mathcal{P}} w_u}.$$

Given the values of the w_u's and c, we determine $\widehat{\boldsymbol{\theta}} = (\widehat{\alpha}, \widehat{\beta})$ by taking derivatives of $\rho(\boldsymbol{\theta}, \rho)$ with respect to each parameter and then setting the resulting gradient equal to zero and solving the system of equations:

$$\widehat{\alpha} = \bar{y}_w - \beta\left(\bar{x}_w - c\right) \quad \text{and} \quad \widehat{\beta} = \frac{\sum_{u\in\mathcal{P}} w_u \left(x_u - \bar{x}_w\right)\left(y_u - \bar{y}_w\right)}{\sum_{u\in\mathcal{P}} w_u \left(x_u - \bar{x}_w\right)^2}$$

where $\bar{y}_w$ and $\bar{x}_w$ are the weighted averages of the y and x values, respectively.

Remark. We could set weights based on the *relative variation* (the ratio between residual standard deviations) of groups. In particular, if `group1.sd / group2.sd` = x, then we could consider setting

$$w_u = \begin{cases} 1 & u \in \text{group 1} \\ x & u \in \text{group 2} \end{cases}$$

[2]The variable c here is chosen to re-center the values of x_u in linear regression. It is commonly chosen to be a meaningful value in the data set, e.g., the average x_u value $c := \bar{x}$. Different choices of c give rise to different interpretations for α; not all such interpretations have practical relevance.

▷ Vector-Valued Attributes: Robust Regression

In WLS, the objective function is modified manually to give influential units (the ones far from the least square line) less weight in the objective function. **Robust regression** has the same goal. The objective function is given by

$$(\widehat{\alpha}, \widehat{\beta}) = \underset{(\alpha,\beta)\in\mathbb{R}^2}{\arg\min} \sum_{u\in\mathcal{P}} \rho\left(y_u - \alpha - \beta\left(x_u - c\right)\right)$$

where different forms of $\rho(\cdot)$ give rise to different fitted lines.

- OLS: Equal weight on each unit.

$$\rho\left(y_u - \alpha - \beta\left(x_u - c\right)\right) = \left[y_u - \alpha - \beta\left(x_u - c\right)\right]^2.$$

- WLS: Less weights to units with LS residuals that are large (in magnitude).

$$\rho\left(y_u - \alpha - \beta\left(x_u - c\right)\right) = w_u\left[y_u - \alpha - \beta\left(x_u - c\right)\right]^2.$$

Here, our goal is to have a function that

- gives lower weight than OLS to units with large residuals;
- quadratic near 0 and hence behaves similarly to OLS for units with small residuals.

The **Huber Loss Function** achieves these goals by combining the quadratic and absolute value functions:

$$\rho_k(r) = \begin{cases} \frac{1}{2}r^2 & \text{for } |r| \leq k \\ k|r| - \frac{1}{2}k^2 & \text{for } |r| > k \end{cases}$$

Below is how to implement this function in R:

```
huber.fn <- function(r, k) {
    val = r^2 / 2
    subr = abs(r) > k
    val[subr] = k * (abs(r[subr]) - k/2)
    return(val)
}
```

An attribute based on this function will be affected by the scale of r, so we might let $k = cS$ where S is a measure of scale. In practice, it is common to use $k \in \{1.345S, 1.5, 2\}$. Note that as k increases, the robust regression with Huber function imposes a larger penalty on larger residuals, hence approaches the OLS fit.

Another form of robust regression involves defining the loss function in terms of *least absolute deviations*:

$$(\widehat{\alpha}, \widehat{\beta}) = \underset{(\alpha,\beta)\in\mathrm{R}^2}{\arg\min} \sum_{u\in\mathcal{P}} \left|y_u - \alpha - \beta\left(x_u - c\right)\right|$$

However, in both Huber and LAD-based regression, the attribute $\widehat{\boldsymbol{\theta}} = (\widehat{\alpha}, \widehat{\beta})$ cannot be solved for a closed form. We need other optimization methods, e.g.,

- gradient descent,
- Newton-Raphson,
- iteratively reweighted least-squares.

These algorithms are employed generally to handle attributes defined implicitly as

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\arg\min}\rho(\boldsymbol{\theta};\mathcal{P})$$

3.2 Gradient Descent

Given an implicitly defined attribute of interest $\boldsymbol{\theta}$, our goal is to construct an *iterative* procedure which produces a sequence of **iterates** $\widehat{\boldsymbol{\theta}_0}, \widehat{\boldsymbol{\theta}}_1, \ldots, \widehat{\boldsymbol{\theta}}_i, \widehat{\boldsymbol{\theta}}_{i+1}, \ldots$ such that this sequence converges to the solution

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\arg\min}\ \rho(\boldsymbol{\theta}; \mathcal{P}).$$

Ideally, each iterate is closer to the optimal solution $\widehat{\boldsymbol{\theta}}$ than the one before it. More specifically, if we are at some point $\widehat{\boldsymbol{\theta}}_i$ on the surface of $\rho(\boldsymbol{\theta}; \mathcal{P})$, we want to move away from $\widehat{\boldsymbol{\theta}}_i$ in the direction that takes us to somewhere on the surface of $\rho(\boldsymbol{\theta}; \mathcal{P})$ that is lower than our current position, i.e.,

$$\rho(\widehat{\boldsymbol{\theta}}_{i+1}; \mathcal{P}) < \rho(\widehat{\boldsymbol{\theta}}_i; \mathcal{P}).$$

From elementary calculus, the direction of this movement is defined by the *gradient* of the surface.

▷ Direction and Step Size

Let $\rho(\boldsymbol{\theta}; \mathcal{P})$ be a *differentiable* function (wrt $\boldsymbol{\theta} \in \mathbb{R}^k$).

The **gradient** of the function wrt $\boldsymbol{\theta}$ is given by

$$\boldsymbol{g} = \boldsymbol{g}(\boldsymbol{\theta}) = \nabla \rho(\boldsymbol{\theta}; \mathcal{P}) = \begin{bmatrix} \dfrac{\partial \rho(\boldsymbol{\theta}; \mathcal{P})}{\partial \theta_1} \\ \dfrac{\partial \rho(\boldsymbol{\theta}; \mathcal{P})}{\partial \theta_2} \\ \vdots \\ \dfrac{\partial \rho(\boldsymbol{\theta}; \mathcal{P})}{\partial \theta_k} \end{bmatrix}$$

To distinguish among the gradient calculations at each iteration, when $\widehat{\boldsymbol{\theta}}_i$ is our best guess at iteration i, we denote the gradient by $\mathbf{g}_i = \mathbf{g}(\widehat{\boldsymbol{\theta}}_i)$. The **normalized gradient**

$$\boldsymbol{d}_i = \frac{\boldsymbol{g}_i}{\|\boldsymbol{g}_i\|}$$

provides the *direction* in which the funtion increases or decreases the fastest. In particular, $\mathbf{d}_i$ indicates the direction of *steepest ascent* and $-\mathbf{d}_i$ indicates the direction of *steepest descent.*

We iterate and obtain a new estimate of $\boldsymbol{\theta}$ by moving in the direction of $-\mathbf{d}_i$ and taking a step of size $\lambda_i > 0$, i.e., the next point is calculated by

$$\widehat{\boldsymbol{\theta}}_{i+1} = \widehat{\boldsymbol{\theta}}_i - \lambda_i \boldsymbol{d}_i.$$

The **step size** λ at each iteration can been chosen in a variety of ways:

- Fixed value, e.g., $\forall i : \lambda_i = 0.1$.
- Fixed sequence, e.g., $\forall i : \lambda_i = 0.1 + 1/i$.
- Perform a *line search* and algorithmically choose the value of λ_i that minimizes

  $$\rho\left(\widehat{\boldsymbol{\theta}}_i - \lambda_i \boldsymbol{d}_i\right)$$

  In other words, we move away from $\widehat{\boldsymbol{\theta}}_i$ in the direction $-\mathbf{d}_i$ minimizing $\rho(\widehat{\boldsymbol{\theta}}_{i+1}; \mathcal{P})$.

To summarize, the **gradient descent** algorithm is as follows. Given some initial value $\widehat{\boldsymbol{\theta}}_0$,

1. Initialize $i \leftarrow 0$.
2. While the sequence of iterates have not converged:
   (a) Calculate the gradient:
   $$\boldsymbol{g}_i = \nabla\rho(\boldsymbol{\theta};\mathcal{P})\big|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}_i}.$$
   (b) Calculate the gradient direction:
   $$\boldsymbol{d}_i \leftarrow \frac{\boldsymbol{g}_i}{\|\boldsymbol{g}_i\|}.$$
   (c) Do a linear search on the step size:
   $$\widehat{\lambda}_i = \arg\min_{\lambda>0} \rho\left(\widehat{\boldsymbol{\theta}}_i - \lambda \boldsymbol{d}_i\right).$$
   (d) Update the iterate:
   $$\widehat{\boldsymbol{\theta}}_{i+1} \leftarrow \widehat{\boldsymbol{\theta}}_i - \widehat{\lambda}_i \boldsymbol{d}_i.$$
   (e) Update the loop variable: $i \leftarrow i+1$.
3. Return $\widehat{\boldsymbol{\theta}} = \widehat{\boldsymbol{\theta}}_i$.

▷ Batch Gradient Descent

In practice, many of the objective functions minimized during statistical analyses have the following form:
$$\rho(\boldsymbol{\theta},\mathcal{P}) = \sum_{u\in\mathcal{P}} \rho(\boldsymbol{\theta};u).$$
The gradient **g** can be expressed as the sum of the unit-specific contributions to the objective function:
$$\boldsymbol{g} = \boldsymbol{g}(\boldsymbol{\theta}) = \nabla\rho(\boldsymbol{\theta};\mathcal{P}) = \sum_{u\in\mathcal{P}} \nabla\rho(\boldsymbol{\theta};u) \equiv \sum_{u\in\mathcal{P}} \boldsymbol{g}(\boldsymbol{\theta};u).$$
Thus, when $\rho(\cdot)$ is a sum over $u \in \mathcal{P}$, the gradient **g** is composed of $N := |\mathcal{P}|$ "smaller" independent gradient calculations and can be arbitrarily grouped/reordered and performed in a parallel fashion. In particular, we can partition $\mathcal{P}$ into H non-overlapping groups (known as **batches**) $\mathcal{P} = \mathcal{B}_1 \cup \cdots \cup \mathcal{B}_H$ each containing M_k units $(k = 1, \ldots, H)$:
$$\boldsymbol{g} = \sum_{u\in\mathcal{P}} \boldsymbol{g}(\boldsymbol{\theta};u) = \sum_{k=1}^{H} \sum_{u\in\mathcal{B}_k} \boldsymbol{g}(\boldsymbol{\theta};u).$$
When computing the gradient **g** is too expensive, we could consider using only a subset of the available data to approximate the actual gradient. In such situations, we typically do not optimize for the step size λ and instead use a fixed step size λ^* to prevent overfitting. Here, λ^* is often referred to as the **learning rate**. Two common approaches are *batch-sequential* and *batch-stochastic* gradient descent.

- *Batch-Sequential.* Sequentially move through H batches and update our estimate $\widehat{\boldsymbol{\theta}}$ after each batch. This differs from ordinary batch gradient descent as the latter updates our estimate only after observing all batches.
- *Batch-Stochastic.* In each iteration, randomly select a batch/sample from the population. Again, the estimate is updated after each batch/sample. When $N = 1$ (batch size), this is known as *stochastic gradient descent.*

3.3 Systems of Equations

Recall that an implicit attribute $\boldsymbol{\theta}$ is the solution to some minimization problem

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\arg\min}\ \rho(\boldsymbol{\theta};\mathcal{P}).$$

In all situations, we need to solve the equation $\nabla\rho(\boldsymbol{\theta};\mathcal{P}) = \mathbf{0}$ to determine the argmin. When the dimension of the vector-valued attribute $\boldsymbol{\theta}$ is k, such an equation is actually a system of k independent equations with k unknowns. Thus, we can also define an implicit $\boldsymbol{\theta}\in\boldsymbol{\Theta}$ as the solution to a system of equations

$$\boldsymbol{\psi}(\boldsymbol{\theta};\mathcal{P}) = \mathbf{0}.$$

In most cases, we can work directly with $\boldsymbol{\psi}(\boldsymbol{\theta};\mathcal{P})$, which usually equals $\nabla\rho(\boldsymbol{\theta};\mathcal{P})$. Unsurpringly, we have

$$\boldsymbol{\psi}(\boldsymbol{\theta};\mathcal{P}) = \sum_{u\in\mathcal{P}}\boldsymbol{\psi}(\boldsymbol{\theta};u),$$

in which case, the attribute of interest $\widehat{\boldsymbol{\theta}}$ is the value that solves

$$\sum_{u\in\mathcal{P}}\boldsymbol{\psi}(\boldsymbol{\theta};u) = \mathbf{0}.$$

▷ **Examples: Scalar-Valued Attributes**

- The *average* is the value of θ that solves $\sum_{u\in\mathcal{P}}\psi(\theta;u) = \sum_{u\in\mathcal{P}} y_u - n\theta = 0$.
- The *weighted average* is the value of θ that solves $\sum_{u\in\mathcal{P}}\psi(\theta;u) = \sum_{u\in\mathcal{P}} w_u\left(y_u-\theta\right) = 0$.
- The *qth quantile* is the smallest value of θ which solves $\sum_{u\in\mathcal{P}}\psi(\theta;u) = \sum_{u\in\mathcal{P}}\frac{1}{N}I\left(y_u\le\theta\right) - q = 0$.

▷ **Examples: Vector-Valued Attributes**

Recall the vector-valued attribute $\boldsymbol{\theta} = (\alpha,\beta)$ in the context of a SLR. Different forms of linear regression arose by determining $\widehat{\boldsymbol{\theta}} = (\widehat{\alpha},\widehat{\beta})$ which was the solution to a variety of different systems of equations:

- *Least squares*:

$$\sum_{u\in\mathcal{P}}\boldsymbol{\psi}(\boldsymbol{\theta};u) = \sum_{u\in\mathcal{P}}\left(y_u-\alpha-\beta\left(x_u-c\right)\right)\begin{pmatrix}1\\ x_u-c\end{pmatrix} = \mathbf{0}.$$

- *Weighted least squares*:

$$\sum_{u\in\mathcal{P}}\boldsymbol{\psi}(\boldsymbol{\theta};u) = \sum_{u\in\mathcal{P}} w_u\left(y_u-\alpha-\beta\left(x_u-c\right)\right)\begin{pmatrix}1\\ x_u-c\end{pmatrix} = \mathbf{0}.$$

- *Robust regression*:

$$\sum_{u\in\mathcal{P}}\boldsymbol{\psi}(\boldsymbol{\theta};u) = \sum_{u\in\mathcal{P}}\rho_k'\left(y_u-\alpha-\beta\left(x_u-c\right)\right)\begin{pmatrix}1\\ x_u-c\end{pmatrix} = \mathbf{0}.$$

The class of methods used to find such solutions to systems of equations are generally referred to as *root finding* methods; that is, it finds x such that $f(x) = 0$.

▷ Newton's Method

Suppose we have a differentiable function $f(x)$ and we wish to find x^* which solves $f(x^*) = 0$. Given an initial x_0, we can use a linear function to approximate $f(x)$ in the vicinity of x_0:

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0).$$

Next, we find the root of the linear approximation to iterate to the next value of x:

$$0 = f(x_0) + f'(x_0)(x - x_0) \quad \Longrightarrow \quad x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

In our case, we want to find $\theta \in \Theta$ such that $\psi(\theta; \mathcal{P}) = 0$. Given the current guess $\hat{\theta} = \hat{\theta}_i$, a first-order approximation is

$$\psi(\theta; \mathcal{P}) \approx \psi(\hat{\theta}_i; \mathcal{P}) + \psi'(\hat{\theta}_i; \mathcal{P}) \cdot (\theta - \hat{\theta}_i)$$

we update the value for the next iteration as

$$\hat{\theta}_{i+1} = \hat{\theta}_i - \frac{\psi(\hat{\theta}_i; \mathcal{P})}{\psi'(\hat{\theta}_i; \mathcal{P})}.$$

Keep going until the algorithm converges.

▷ Newton-Raphson Method

The multi-variate analog of Newton's method is the **Newton-Raphson** method. We wish to solve

$$\boldsymbol{\psi}(\boldsymbol{\theta}; \mathcal{P}) = \mathbf{0}.$$

Viewing $\boldsymbol{\psi}(\boldsymbol{\theta}; \mathcal{P})$ as a differentiable $k \times 1$ vector $(\psi_1, \psi_2, \ldots, \psi_k)^T$, we care about the Jacobian matrix:

$$\boldsymbol{\psi}'(\boldsymbol{\theta}; \mathcal{P}) = \frac{\partial \boldsymbol{\psi}(\boldsymbol{\theta}; \mathcal{P})}{\partial \boldsymbol{\theta}} = \begin{bmatrix} \frac{\partial \psi_1}{\partial \theta_1} & \cdots & \frac{\partial \psi_1}{\partial \theta_k} \\ \vdots & \ddots & \vdots \\ \frac{\partial \psi_k}{\partial \theta_1} & \cdots & \frac{\partial \psi_k}{\partial \theta_k} \end{bmatrix}.$$

Given a current guess $\widehat{\boldsymbol{\theta}}_i$, we can use a linear function to approximate the function $\boldsymbol{\psi}(\boldsymbol{\theta}; \mathcal{P})$. A first-order approximation of $\boldsymbol{\psi}(\boldsymbol{\theta}; \mathcal{P})$ in the vicinity of $\widehat{\boldsymbol{\theta}}_i$ can be written as

$$\boldsymbol{\psi}(\boldsymbol{\theta}; \mathcal{P}) \approx \boldsymbol{\psi}(\widehat{\boldsymbol{\theta}}_i; \mathcal{P}) + \boldsymbol{\psi}'(\widehat{\boldsymbol{\theta}}_i; \mathcal{P}) \cdot (\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_i).$$

Rearranging, the vector at which the linear approximation is equal to zero is

$$\boldsymbol{\theta} \approx \widehat{\boldsymbol{\theta}}_i - [\boldsymbol{\psi}'(\widehat{\boldsymbol{\theta}}_i; \mathcal{P})]^{-1} \boldsymbol{\psi}(\widehat{\boldsymbol{\theta}}_i; \mathcal{P}) = \widehat{\boldsymbol{\theta}}_{i+1} = \widehat{\boldsymbol{\theta}}_i - H^{-1} \boldsymbol{g}_i.$$

Keep going until the algorithm converges.

Remark. When the objective function is $\rho(\boldsymbol{\theta}; \mathcal{P})$, then $\boldsymbol{\psi}(\boldsymbol{\theta}; \mathcal{P}) = \nabla \rho(\boldsymbol{\theta}; \mathcal{P}) = \boldsymbol{g}$ and the system $\boldsymbol{\psi}(\boldsymbol{\theta}; \mathcal{P}) = \mathbf{0}$ is equivalent to $\nabla \rho(\boldsymbol{\theta}; \mathcal{P}) = \mathbf{0}$. Therefore, we have $\boldsymbol{\psi}'(\boldsymbol{\theta}; \mathcal{P}) = H^{-1}$ where H denotes the Hessian of ρ (the matrix of second-order partial derivatives of ρ wrt $\theta_1, \ldots, \theta_k$).

Remark. Recall that the updating equation associated with gradient descent is given by

$$\widehat{\boldsymbol{\theta}}_{i+1} = \widehat{\boldsymbol{\theta}}_i - \widehat{\lambda}_i \boldsymbol{d}_i = \widehat{\boldsymbol{\theta}}_i - \frac{\widehat{\lambda}_i}{\|\boldsymbol{g}_i\|} \boldsymbol{g}_i.$$

Thus, NR does the same thing with the step sizes modulated by the Hessian of the objective function.

▷ Iteratively Reweighted Least Squares

IRLS provides an iterative algorithm for finding $\widehat{\boldsymbol{\theta}} = (\widehat{\alpha}, \widehat{\beta})'$, the solution to

$$\underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\arg\min} \sum_{u\in\mathcal{P}} \rho\left(y_u - \alpha - \beta\left(x_u - c\right)\right).$$

Differentiating wrt α and β and setting the result equal to zero yields

$$\sum_{u\in\mathcal{P}} \rho'\left(y_u - \alpha - \beta\left(x_u - c\right)\right)\begin{pmatrix} 1 \\ x_u - c \end{pmatrix} = \mathbf{0}.$$

Setting $r_u = y_u - \alpha - \beta(x_u - c)$ and $z_u = (1, x_u - c)^T$, so the system above becomes

$$\sum_{u\in\mathcal{P}} \rho'\left(r_u\right)\boldsymbol{z}_u = \mathbf{0}.$$

In the case of weighted least squares where $\rho(r_u) = w_u r_u^2$ and $\rho'(r) = 2w_u r_u$, the system reduces to

$$\sum_{u\in\mathcal{P}} w_u r_u \boldsymbol{z}_u = \mathbf{0}.$$

Now exploit the fact that the general objective function minimization problem can be recasted as a weighted least squares problem:

$$\begin{aligned}
\mathbf{0} &= \sum_{u\in\mathcal{P}} \rho'\left(r_u\right)\boldsymbol{z}_u \\
&= \sum_{u\in\mathcal{P}} \left(\frac{\rho'\left(r_u\right)}{r_u}\right) r_u \boldsymbol{z}_u \\
&= \sum_{u\in\mathcal{P}} w_u r_u \boldsymbol{z}_u
\end{aligned}$$

where $w_u = \rho'(r_u)/r_u$ is the weight for unit u provided that $r_u \neq 0$. We want this transformation because WLS has a closed form solution, as discussed below.

If we had some initial value for the residuals r_u (and hence the weights w_u), we could solve this equation in closed form yielding a value for $\boldsymbol{\theta}$, call it $\widehat{\boldsymbol{\theta}}_1 = (\widehat{\alpha}_1, \widehat{\beta}_1)'$. Since $\boldsymbol{\theta}_0 = (\alpha_0, \beta_0)$ is likely far from $\widehat{\boldsymbol{\theta}} = (\hat{\alpha}, \hat{\beta})^T$, we should iterate and update our values of the residuals (and hence weights) with $\widehat{\boldsymbol{\theta}}_1 = (\widehat{\alpha}_1, \widehat{\beta}_1)'$. Repeat this process until the estimates of the parameters converge. This process is IRLS. Let's see the algorithm.

1. Initialize $i \leftarrow 0$.
2. While not converaged:
   (a) Construct residuals and weights for all $u \in \mathcal{P}$:
   $$r_u = y_u - \boldsymbol{z}_u'\boldsymbol{\theta}_i, \quad w_u = \frac{\rho'(r_u)}{r_u}.$$
   (b) Solve the weighted leaset squares problem, i.e., find $\widehat{\boldsymbol{\theta}}$, the value of $\boldsymbol{\theta}$ such that
   $$\sum_{u\in\mathcal{P}} w_u\left(y_u - \boldsymbol{z}_u'\boldsymbol{\theta}\right)\boldsymbol{z}_u = \mathbf{0}.$$
   (c) Update the parameter $\widehat{\boldsymbol{\theta}}_{i+1} \leftarrow \widehat{\boldsymbol{\theta}}$.
3. Return $\widehat{\boldsymbol{\theta}} = \widehat{\boldsymbol{\theta}}_i$.