Data Analysis SIT718

Delaram Pahlevani

weighted arithmetic mean (WAM)

The weighted arithmetic mean is similar to an ordinary arithmetic mean (the most common type of average), except that instead of each of the data points contributing equally to the final average, some data points contribute more than others. The notion of weighted mean plays a role in descriptive statistics and also occurs in a more general form in several other areas of mathematics.

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i},$$

Yezi : 6.3
Jimin: 6.6
Hyolyn: 6.6

Candidate	Judge 1	Judge 2	Judge 3
Yezi	9	6	4
Jimin	7	7	6
Hyolyn	4	8	8

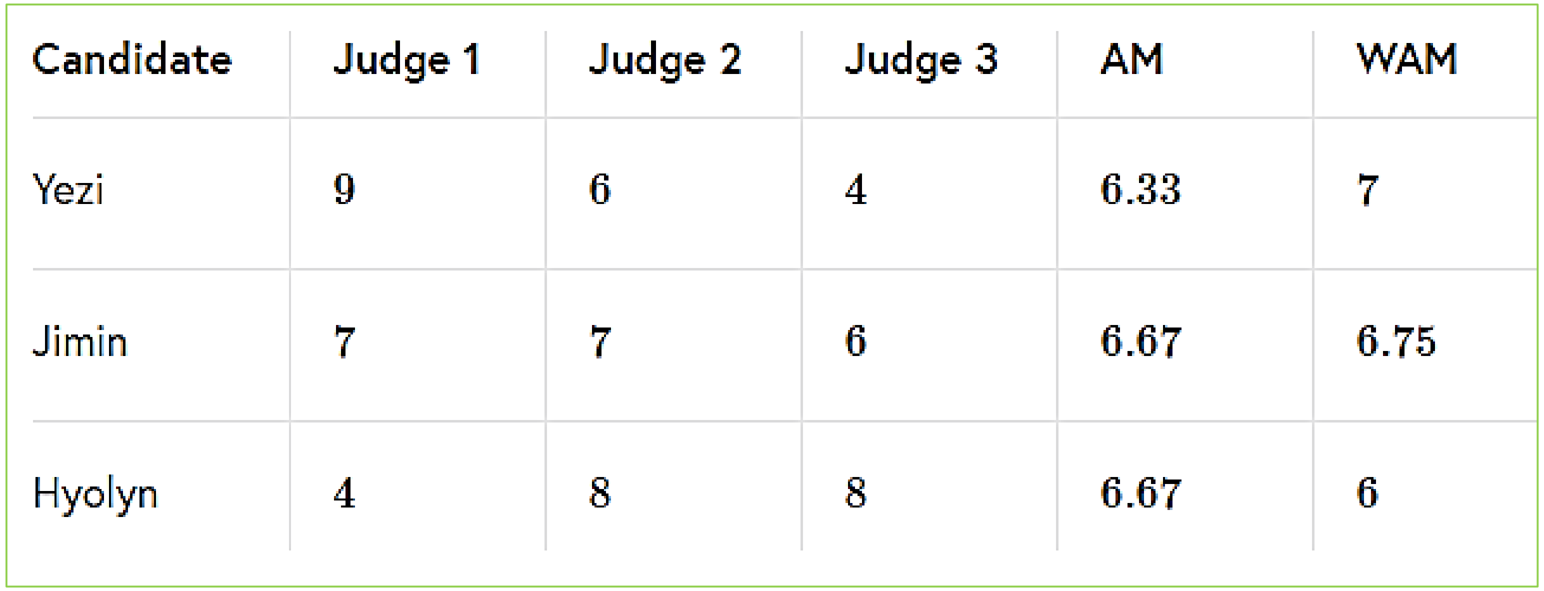
weighted arithmetic mean (WAM)

EXAMPLE: Suppose we have candidates being evaluated by three ‘judges’ for an internship position. Each judge provides a score which is then combined into an overall score for each candidate. If we assume each judge’s score has equal importance, then calculate the arithmetic mean :

$$\mathrm{AM} = \frac{1}{3}(x_1 + x_2 + x_3) = \frac{1}{3}x_1 + \frac{1}{3}x_2 + \frac{1}{3}x_3 .$$

weighted arithmetic mean (WAM)

EXAMPLE: Results using the both arithmetic mean and weighted arithmetic mean are shown below



Candidate	Judge 1	Judge 2	Judge 3	AM	WAM
Yezi	9	6	4	6.33	7
Jimin	7	7	6	6.67	6.75
Hyolyn	4	8	8	6.67	6

weighted arithmetic mean (WAM)

EXAMPLE: If Judge 1 is a manager whose decision has more importance, we can instead use the coefficients 1/2, 1/4, 1/4 or 0.5, 0.25, 0.25, resulting in the weighted arithmetic mean (WAM) calculation:

$$\mathrm{WAM} = \frac{1}{2}x_1 + \frac{1}{4}x_2 + \frac{1}{4}x_3.$$

```
#Weighted Arithmetic Mean
WAM <- function(x= c(0.6, 0.7, 0.1),w= c(0.5,0.2,0.3)){
   sum(w*x)
 }
WAM(x,w)
```

weighted arithmetic mean (WAM)

EXAMPLE - R: The illustrated example can be coded in R as below:

```
#motivation example
candidate <- read.csv("judge example.csv")
View(candidate)
weight1=0.20
weight2=0.5
weight3= 0.25
candidateScores = function(weight1, weight2, weight3) {
  if(weight1+weight2+weight3 == 1) {
    YeziScore = 9*weight1 + 6*weight2 + 4*weight3
    JiminScore = 7*weight1 + 7*weight2 + 6*weight3
    HyolynScore = 4*weight1 + 8*weight2 + 8*weight3
    print(paste("Yezi's Score: ", YeziScore))
    print(paste("Jimin's Score: ", JiminScore))
    print(paste("Hyolyn's Score: ", HyolynScore))
  } else {    print("Your weights do not add up to 1.") }
}
candidateScores(weight1, weight2, weight3)
```

weighted arithmetic mean (WAM)

EXAMPLE : the vector of parameters which we denote as w=(w1,w2,…,wn) is often referred to as a weighting vector. To keep them in line with aggregation functions, the weights should satisfy the following conditions:

1. All normalised weights must be greater than or equal to zero i.e. wi≥0 for all I

2. adding up all weights must equal 1 $\sum_{i=1}^{n} w_i = 1$

weighted arithmetic mean (WAM)

EXAMPLE : Evaluate the weighted arithmetic mean when W = (0.5, 0.2, 0.3) and the input x = (0.6, 0.7, 0.1)?

$$WAM_{\mathbf{w}}(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

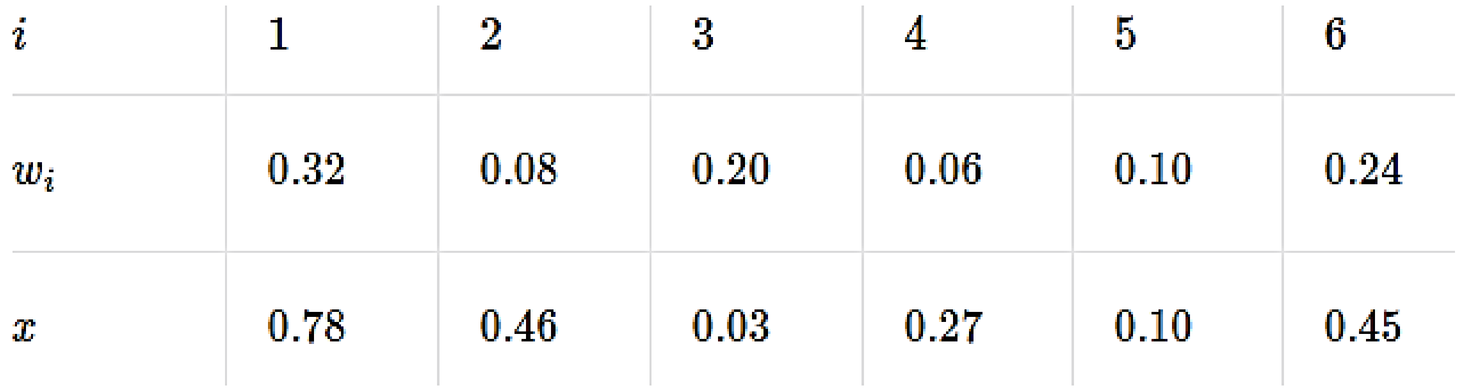
$$WAM = (0.5 \times 0.6) + (0.2 \times 0.7) + (0.3 \times 0.1) = 0.47$$

```
#Weighted Arithmetic Mean
WAM <- function(x,w){
  sum(w*x)
}
x= c(0.6, 0.7, 0.1)
w= c(0.5,0.2,0.3)
WAM(x,w)
```

Weighted median

The standard median is the middle value when all of the inputs are arranged in order. In the case of the weighted median, each input might be associated with a varying degree of importance or density.

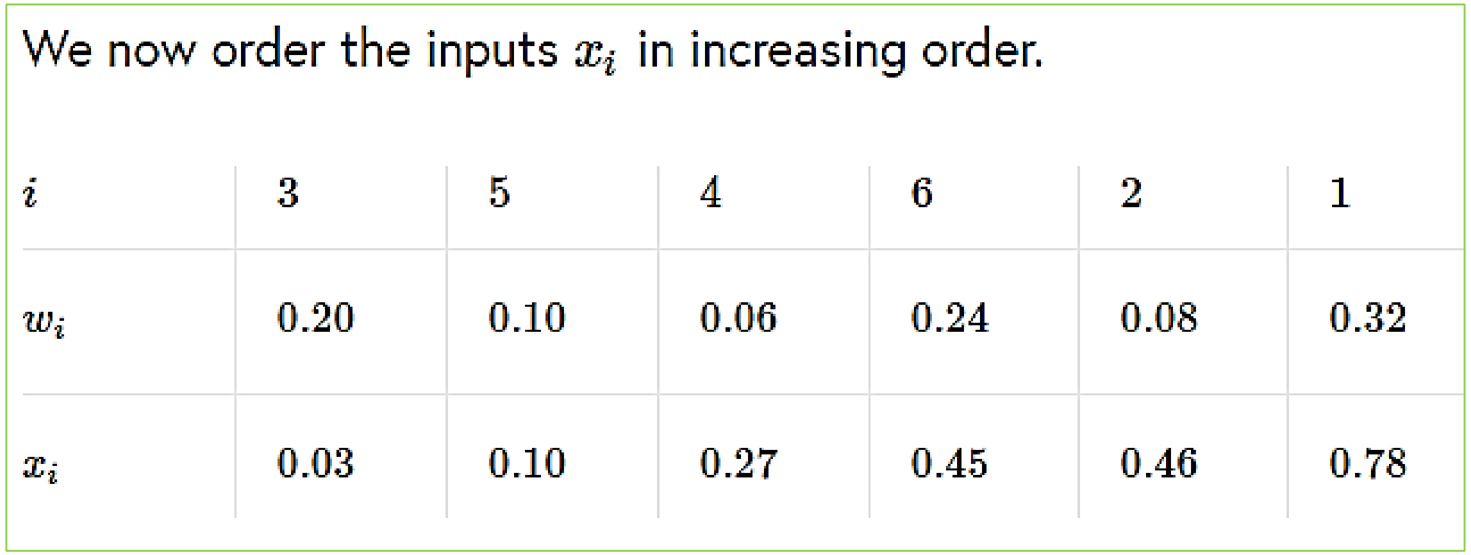
EXAMPLE : Let w = (0.32,0.08,0.20,0.06,0.1,0.24) and x = (0.78,0.45,0.03,0.27,0.1,0.45).



i	1	2	3	4	5	6
w_i	0.32	0.08	0.20	0.06	0.10	0.24
x	0.78	0.46	0.03	0.27	0.10	0.45

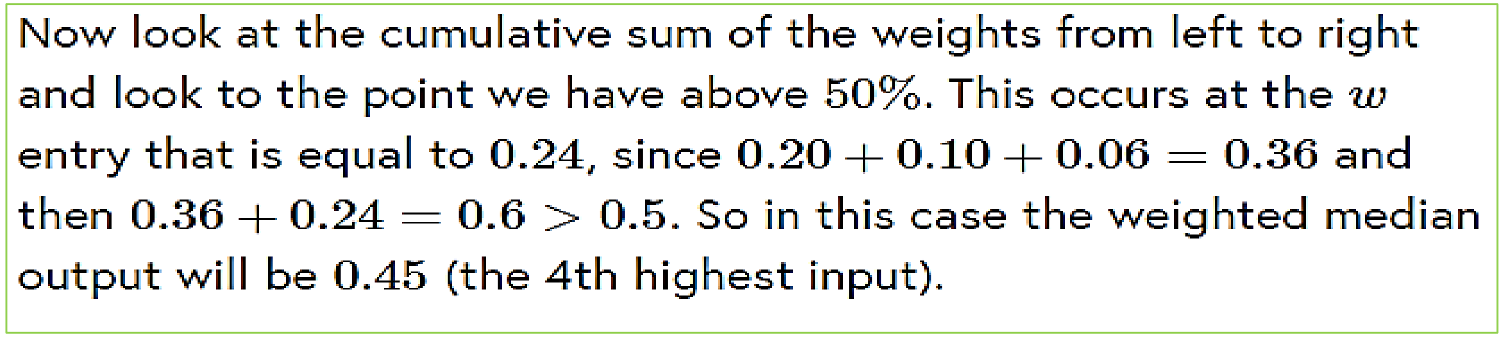
Weighted median

EXAMPLE :



We now order the inputs x_i in increasing order.

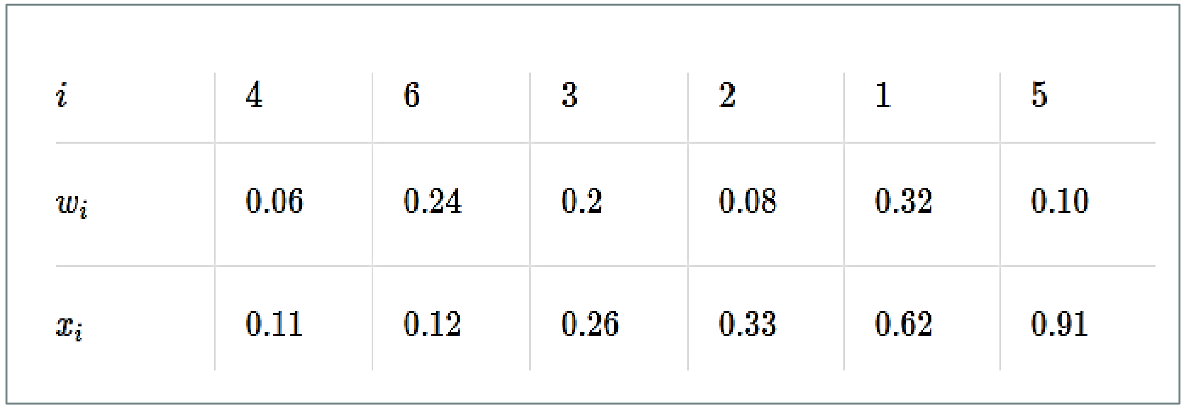
i	3	5	4	6	2	1
w_i	0.20	0.10	0.06	0.24	0.08	0.32
x_i	0.03	0.10	0.27	0.45	0.46	0.78



Now look at the cumulative sum of the weights from left to right and look to the point we have above 50%. This occurs at the w entry that is equal to 0.24, since $0.20 + 0.10 + 0.06 = 0.36$ and then $0.36 + 0.24 = 0.6 > 0.5$. So in this case the weighted median output will be 0.45 (the 4th highest input).

Weighted median (What if we exactly have 50%?)

EXAMPLE :



i	4	6	3	2	1	5
w_i	0.06	0.24	0.2	0.08	0.32	0.10
x_i	0.11	0.12	0.26	0.33	0.62	0.91

- We can either take the half-way point between the two values on the 50% border ($(0.26 + 0.33)/2 = 0.295$).
- Or we can use the lower value of 0.26 (which is referred to as the 'lower weighted median') or we can take the higher value of 0.33 (referred to as the 'upper weighted median').

Weighted median (What if we exactly have 50%?)

EXAMPLE :

```
#Weighted median

Wmed <- function(x,w) {
  w <- w/sum(w)
  n <- length(x)
  w <- w[order(x)]
  x <- sort(x)
  out <- x[1]
  for(i in 1:(n-1)) {
    if(sum(w[1:i]) < 0.5)
out <- x[i+1]
  }
```

```
# input
w1 <- c(0.32, 0.08, 0.20, 0.06, 0.10, 0.24)
x1 <- c(0.78, 0.45, 0.03, 0.27, 0.10, 0.45)
Wmed(x,w)
# Result should be 0.45
w2 <- c(0.1, 0.4, 0.3, 0.2)
x2 <- c(0.3, 0.7, 0.8, 0.2)
# Result should be 0.7
```

Trimmed Mean

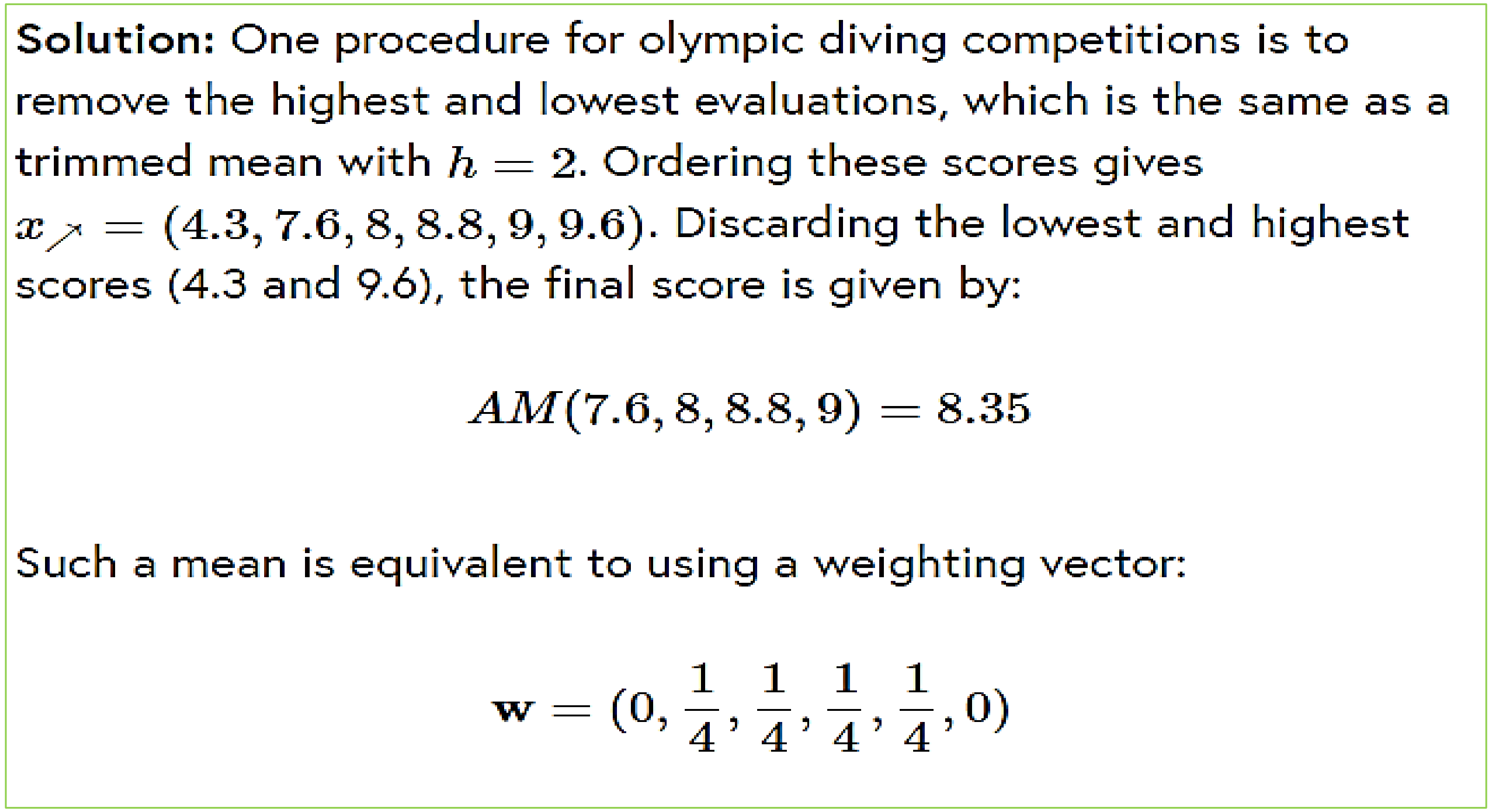
a robust measurement that is a compromise between the arithmetic mean and the median to help us deal with outliers.

The trimmed mean discards the highest *h/2* inputs and the lowest *h/2* inputs and then takes the average of those remaining. As a weighting vector, this would be represented by:

$$w = (\underbrace{0, \ldots, 0}_{\frac{h}{2}\text{ zeros}}, \frac{1}{n-h}, \frac{1}{n-h}, \ldots, \frac{1}{n-h}, \underbrace{0, \ldots, 0}_{\frac{h}{2}\text{ zeros}})$$

Trimmed Mean

EXAMPLE : A diver receives the scores x=(9,8.8,9.6,4.3,7.6,8) for one of their dives. What will be the final score?



Solution: One procedure for olympic diving competitions is to remove the highest and lowest evaluations, which is the same as a trimmed mean with $h = 2$. Ordering these scores gives $x_{\nearrow} = (4.3, 7.6, 8, 8.8, 9, 9.6)$. Discarding the lowest and highest scores (4.3 and 9.6), the final score is given by:

$$AM(7.6, 8, 8.8, 9) = 8.35$$

Such a mean is equivalent to using a weighting vector:

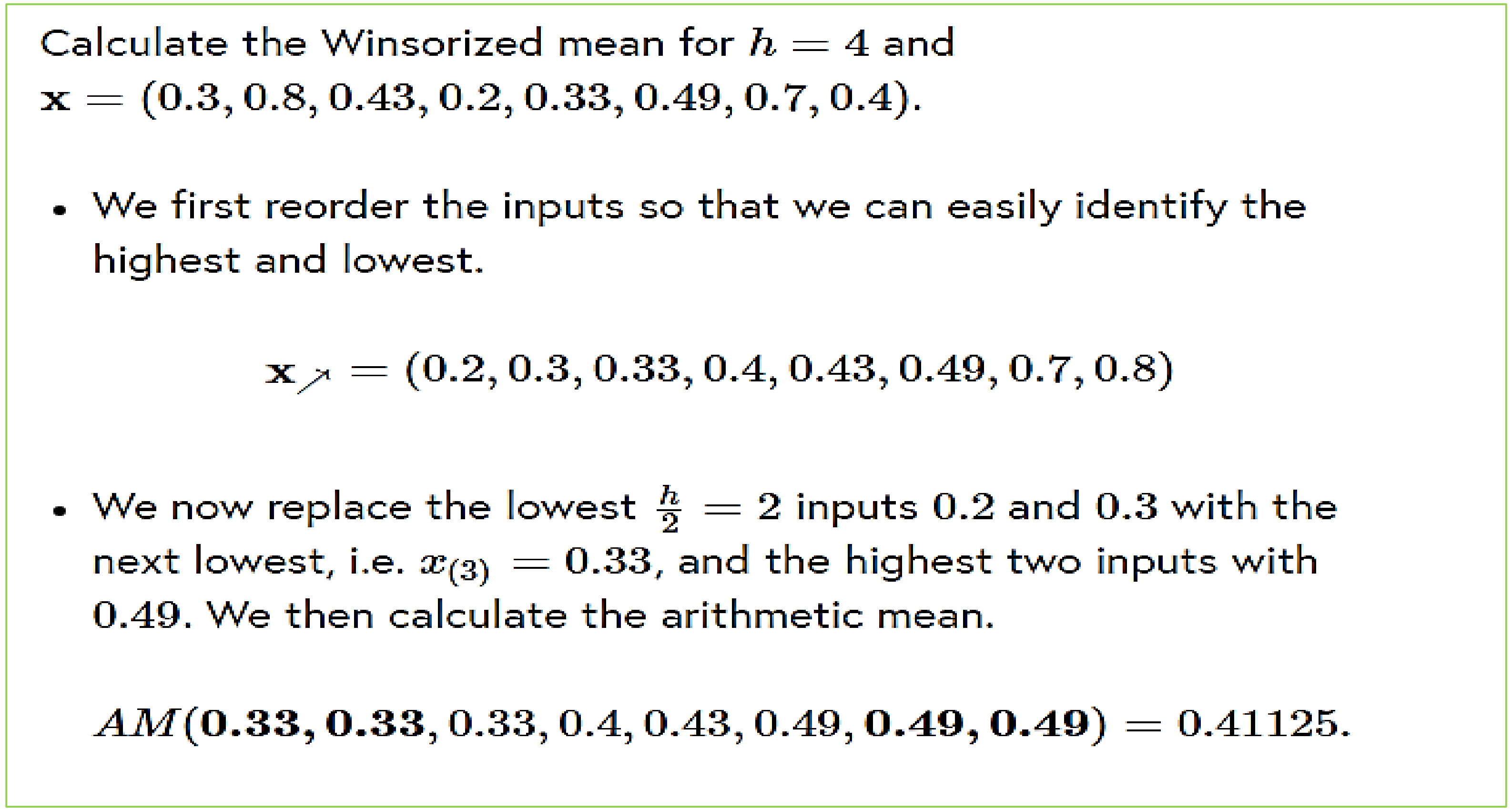
$$\mathbf{w} = (0, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, 0)$$

```
x<-c(9, 8.8, 9.6, 4.3, 7.6, 8)
AM1=mean(x)
AM1
x1<-c(9,8.8,7.6,8)
AM2=mean(x1)
AM2
mean(x, trim = 1/6)
```

Winsorized Mean

The Winsorized mean is similar in that the highest and lowest *h/2* of inputs are replaced with the next highest/lowest of the inputs.

EXAMPLE :



Calculate the Winsorized mean for $h = 4$ and $\mathbf{x} = (0.3, 0.8, 0.43, 0.2, 0.33, 0.49, 0.7, 0.4)$.

- We first reorder the inputs so that we can easily identify the highest and lowest.

$$\mathbf{x}_{\nearrow} = (0.2, 0.3, 0.33, 0.4, 0.43, 0.49, 0.7, 0.8)$$

- We now replace the lowest $\frac{h}{2} = 2$ inputs 0.2 and 0.3 with the next lowest, i.e. $x_{(3)} = 0.33$, and the highest two inputs with 0.49. We then calculate the arithmetic mean.

$$AM(\mathbf{0.33}, \mathbf{0.33}, 0.33, 0.4, 0.43, 0.49, \mathbf{0.49}, \mathbf{0.49}) = 0.41125.$$

Winsorized Mean

EXAMPLE :

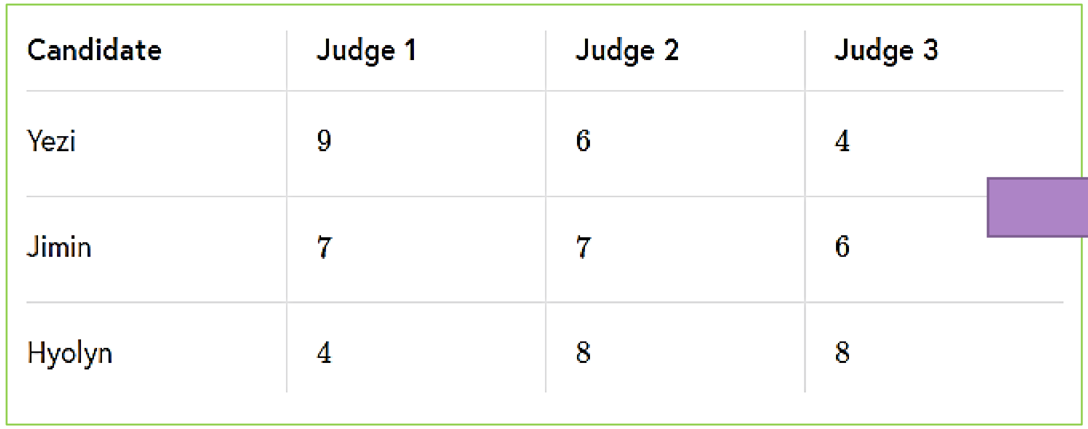
```
#winsorized mean
Wins.mean <- function(x,h=0) {
  n <- length(x)
  repl <- floor(h*n)
  x <- sort(x)
  x[1:repl] <- x[repl+1]
  x[(n-repl+1):n] <- x[n-repl]
  mean(x)
}
```

```
# Winsorized Mean
wData <- c(0.2, 0.3, 0.33, 0.4, 0.43, 0.49, 0.7, 0.8)
wPercentage <- 0.25
Wins.mean(wData, wPercentage)
# Expected Result: 0.41125
```

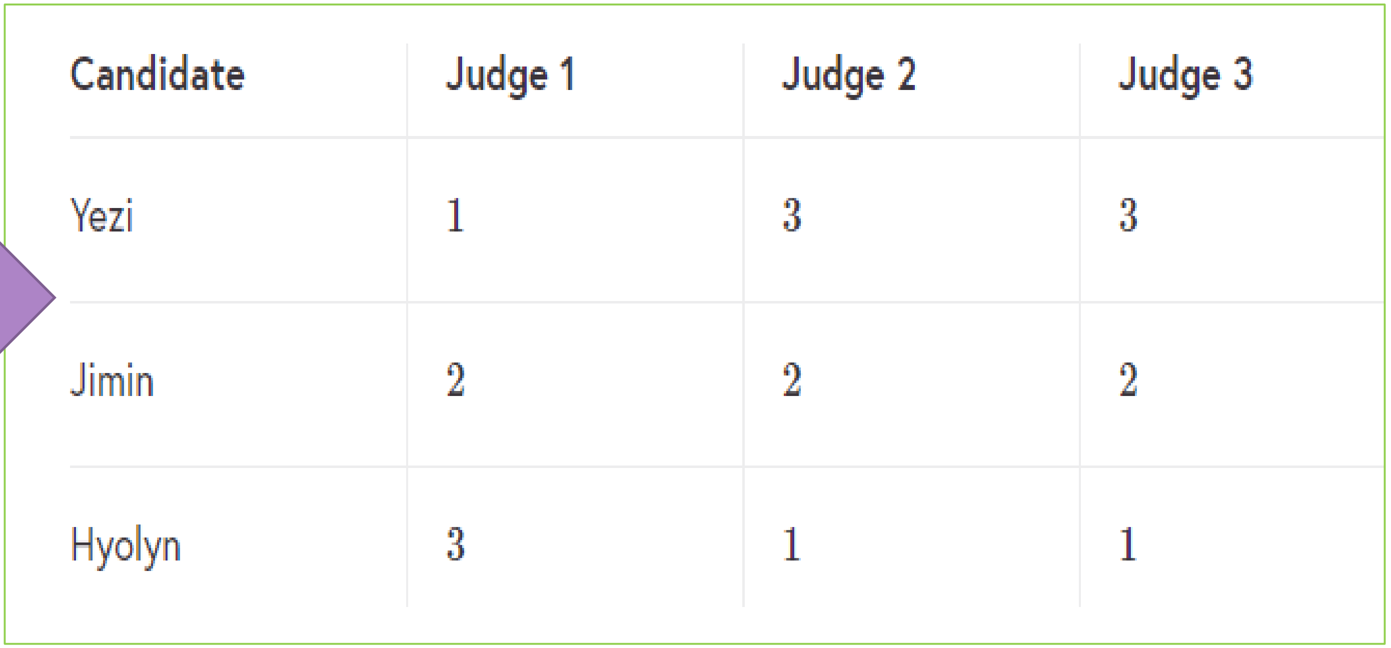
Borda Count

Instead of calculating the weighted average of individual scores, what if we replaced them with rankings?
Now we rank candidates based on their scores.

EXAMPLE :



Candidate	Judge 1	Judge 2	Judge 3
Yezi	9	6	4
Jimin	7	7	6
Hyolyn	4	8	8



Candidate	Judge 1	Judge 2	Judge 3
Yezi	1	3	3
Jimin	2	2	2
Hyolyn	3	1	1

Borda Count

EXAMPLE :

Candidate	# 1st Place	# 2nd Place	# 3rd Place	Score
Yezi	2	0	0	2+0+0=2
Jimin	1	1	1	1+1+1=3
Hyolyn	0	2	2	0+2+2=4

Borda Count example

```
Borda<-function(x,w){
 total.scores<-array(0,nrow(x))
 for(j in 1:ncol(x)){
  x[,j]<-rank(x[,j])
 }
 for(i in 1:length(w)){
  x[x==i]<-w[i]
 }
 for(i in 1:nrow(x)){
  total.scores[i]<-sum(x[i,])
 }
 total.scores
}
x <- array(c(9,7,4,6,7,8,4,6,8), c(3,3))
w <- c(0,1,2)
Borda(x,w)
```

Simpson’s Dominance Index

Measures the degree of concentration when individuals are classified into types.[

$$q = \frac{x_i}{\sum_{i=1}^{n} x_i}. \quad \Rightarrow \quad Simp(\mathbf{q}) = \sum_{i=1}^{n} q_i^2$$

For example, if there are four species of birds and their populations in a particular region are given by $\mathbf{x} = (123, 256, 309, 777)$, then as proportions of the total number of birds we would have $\mathbf{q} = (0.084, 0.175, 0.211, 0.530)$.

Entropy

similar to the geometric mean however without the inverse performed at the end.

$$\text{Entropy}(\mathbf{q}) = -\sum_{i=1}^{n} q_i \ln q_i$$

Weighted Power mean

$$PM_{\mathbf{w}}(\mathbf{x}) = \left(\sum_{i=1}^{n} w_i x_i^p\right)^{\frac{1}{p}} = (w_1 x_1^p + w_2 x_2^p + \cdots + w_n x_n^p)^{\frac{1}{p}}$$

Recall that for the **power mean**:

$p \to -\infty$	**Minimum**
$p = -1$	**Harmonic mean**
$p = 0$	**Geometric mean**
$p = 1$	**Arithmetic Mean**
$p = 2$	**Quadratic mean**
$p \to \infty$	**Maximum**

```
#weighted power mean
WPM <- function(x,w,p) {
  if(p == 0) {prod(x^w)}
  else {sum(w*(x^p))^(1/p)}
}
```

```
# Weighted Power Means (Geometric
Mean) Example
w <- c(0.5, 0.2, 0.3)
x1 <- c(0.7, 0.6, 0.3)
x2 <- c(0.8, 0.6, 0.3)
x3 <- c(0.7, 0.6, 0.4)
WPM(x, w, 0)
# Results: 0.5264, 0.5627, 0.5738
```

Ordered Weighted Averaging (OWA)

weights are applied *after* the scores are ordered.

$$OWA_{\mathbf{w}}(\mathbf{x}) = \sum_{i=1}^{n} w_i x_{(i)} = w_1 x_{(1)} + w_2 x_{(2)} + \cdots + w_n x_{(n)}$$

EXAMPLE :

Suppose $\mathbf{x} = (0.3, 0.6, 0.8, 0.2)$ and $\mathbf{w} = (0.1, 0.3, 0.4, 0.2)$.
Ordering $\mathbf{x}$ gives $x_{\nearrow} = (0.2, 0.3, 0.6, 0.8)$ and thus

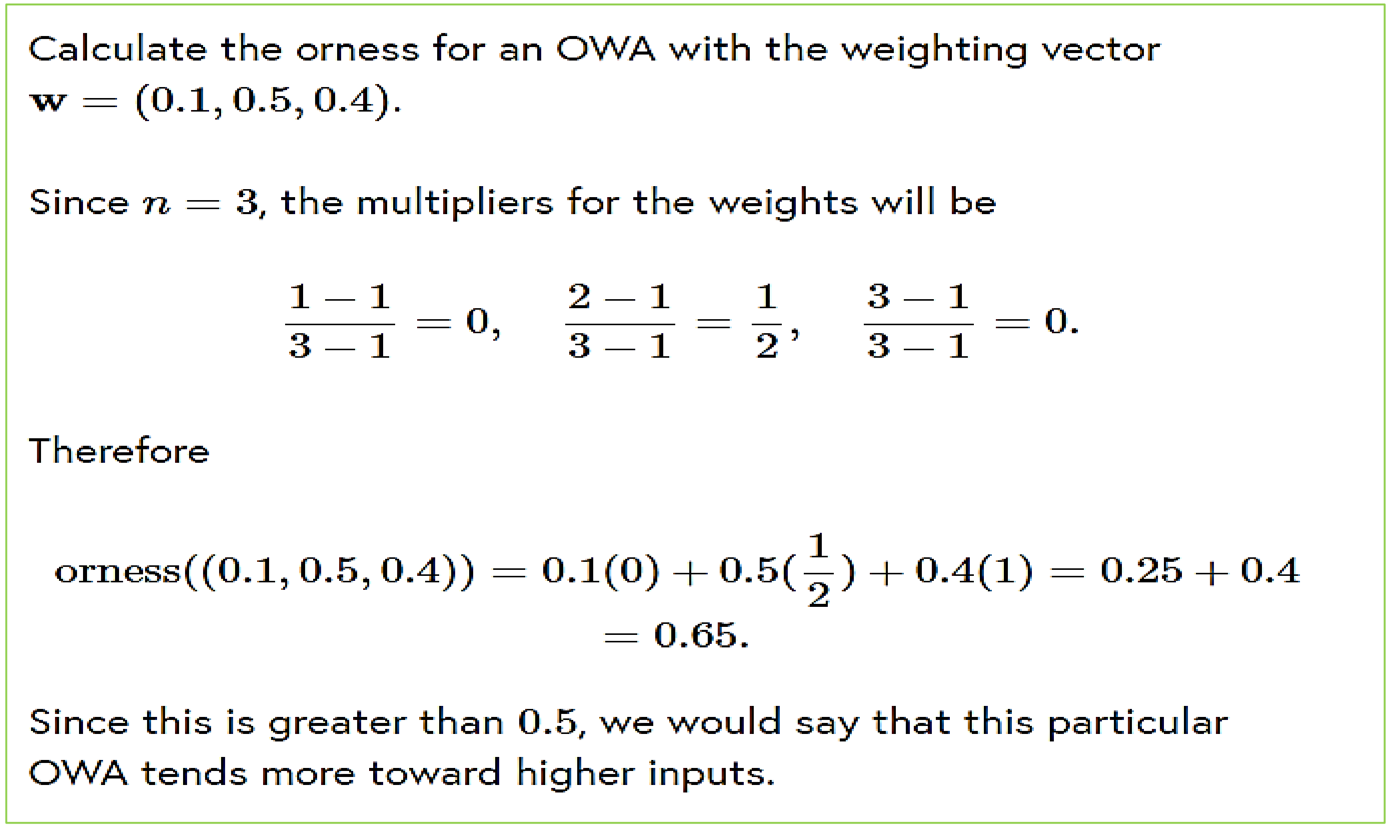
$$OWA_{\mathbf{w}}(\mathbf{x}) = (0.1 \times 0.2) + (0.3 \times 0.3) + (0.4 \times 0.6) + (0.2 \times 0.8) = 0.51$$

Orness

weights are applied *after* the scores are ordered.

$$\sum_{i=1}^{n} w_i \frac{i-1}{n-1}$$

EXAMPLE :



Calculate the orness for an OWA with the weighting vector $\mathbf{w} = (0.1, 0.5, 0.4)$.

Since $n = 3$, the multipliers for the weights will be

$$\frac{1-1}{3-1} = 0, \quad \frac{2-1}{3-1} = \frac{1}{2}, \quad \frac{3-1}{3-1} = 0.$$

Therefore

$$\text{orness}((0.1, 0.5, 0.4)) = 0.1(0) + 0.5(\frac{1}{2}) + 0.4(1) = 0.25 + 0.4$$
$$= 0.65.$$

Since this is greater than 0.5, we would say that this particular OWA tends more toward higher inputs.

Orness

weights are applied *after* the scores are ordered.

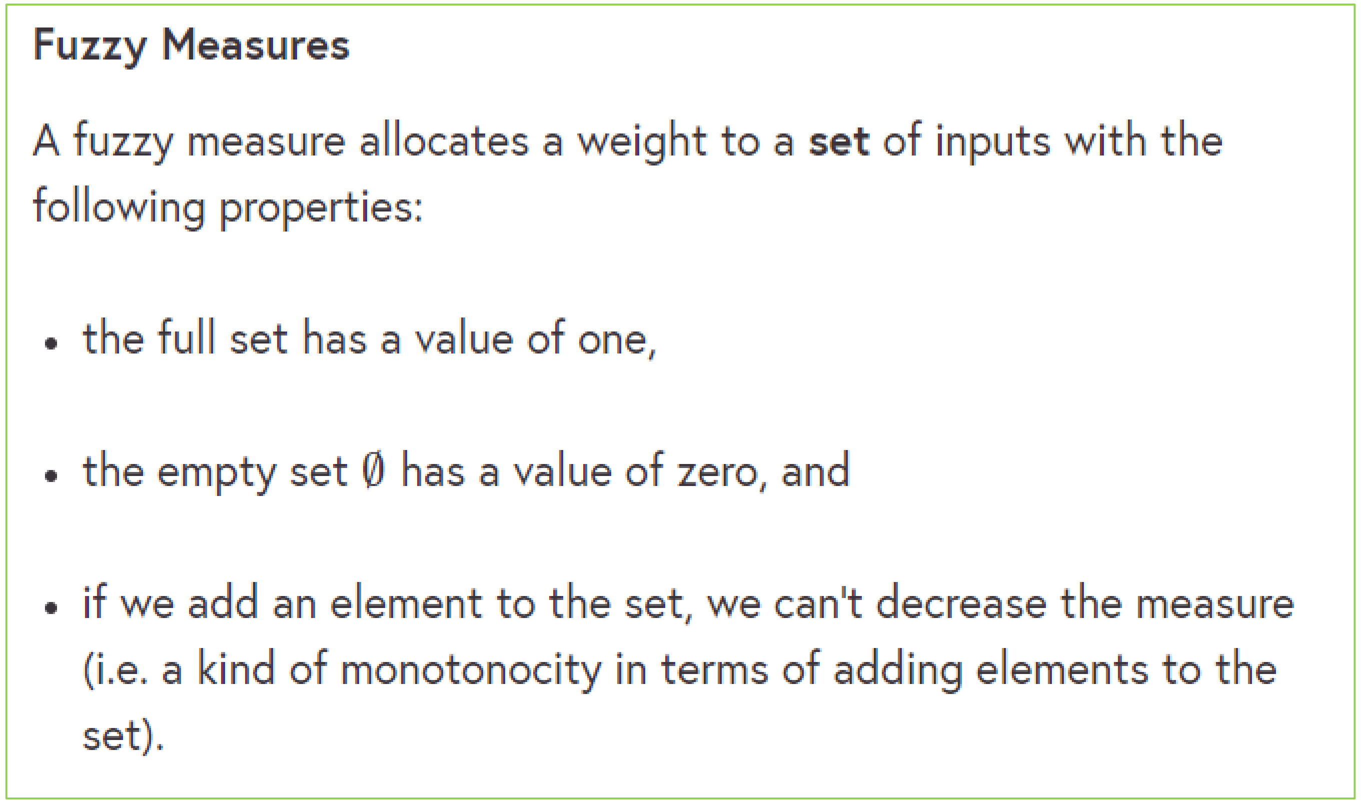
EXAMPLE :

```
#Orness
orness.OWA = function(w) {
  n = length(w)
  sum(w*(1:n-1)/(n-1))
}
orness.OWA(w)
```

```
# Orness Example 1
ornessWeights <- c(0.1, 0.5, 0.4)
ornessWeights2 <- c(0.1, 0.2, 0.2, 0.4,
0.1)# Expected Result: 0.65, 0.55
```

Fuzzy Measures

We now turn to a very useful operator that applies weights to arguments based on both their source (weighted power mean) and their relative size (OWA).

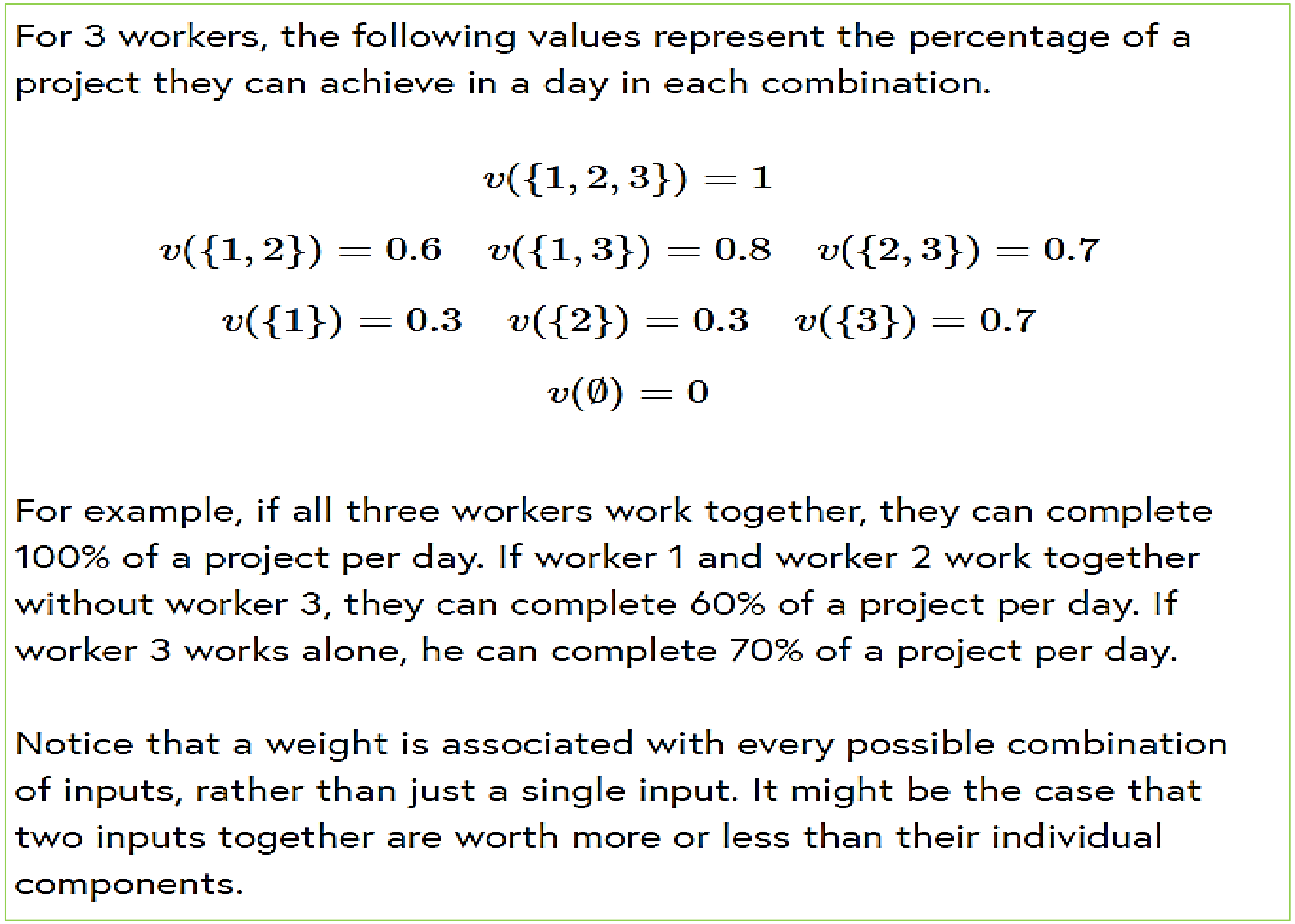


Fuzzy Measures

A fuzzy measure allocates a weight to a **set** of inputs with the following properties:

- the full set has a value of one,
- the empty set ∅ has a value of zero, and
- if we add an element to the set, we can't decrease the measure (i.e. a kind of monotonocity in terms of adding elements to the set).

Fuzzy Measures



For 3 workers, the following values represent the percentage of a project they can achieve in a day in each combination.

$$v(\{1,2,3\}) = 1$$

$$v(\{1,2\}) = 0.6 \quad v(\{1,3\}) = 0.8 \quad v(\{2,3\}) = 0.7$$

$$v(\{1\}) = 0.3 \quad v(\{2\}) = 0.3 \quad v(\{3\}) = 0.7$$

$$v(\emptyset) = 0$$

For example, if all three workers work together, they can complete 100% of a project per day. If worker 1 and worker 2 work together without worker 3, they can complete 60% of a project per day. If worker 3 works alone, he can complete 70% of a project per day.

Notice that a weight is associated with every possible combination of inputs, rather than just a single input. It might be the case that two inputs together are worth more or less than their individual components.

Choquet Integral

- Choquet Integral is an aggregation function defined with respect to the fuzzy measure. A fuzzy measure is a set function, acting on the domain of all possible combinations of a set of criteria. The complexity is therefore exponential of 2n subsets, where n is the number of criteria.

- So far we have looked at applying weights to arguments either depending on their source (weighted power means) or depending on their relative size (the OWA). We now turn to a very useful (but a somewhat complex) operator that can take both into account.

Choquet Integral

$$C_v(\mathbf{x}) = \sum_{i=1}^{n} x_{[i]} \left(v(\{[i],[i+1],\ldots,[n]\}) - v(\{[i+1],[i+2],\ldots,[n]\})\right)$$

Let's work through a relatively simple example with $\mathbf{x} = \langle 0.3, 0.7, 0.2 \rangle$ and the values of the fuzzy measure given as follows.

$$v(\{1,2,3\}) = 1$$

$$v(\{1,2\}) = 0.4 \quad v(\{1,3\}) = 0.9 \quad v(\{2,3\}) = 0.5$$

$$v(\{1\}) = 0.3 \qquad v(\{2\}) = 0.4 \qquad v(\{3\}) = 0.1$$

$$v(\emptyset) = 0$$

Choquet Integral

$$C_v(\mathbf{x}) = \sum_{i=1}^{n} x_{[i]} \big(v(\{[i], [i+1], \ldots, [n]\}) - v(\{[i+1], [i+2], \ldots, [n]\})\big)$$

Let's work through a relatively simple example with $\mathbf{x} = \langle 0.3, 0.7, 0.2 \rangle$ and the values of the fuzzy measure given as follows.

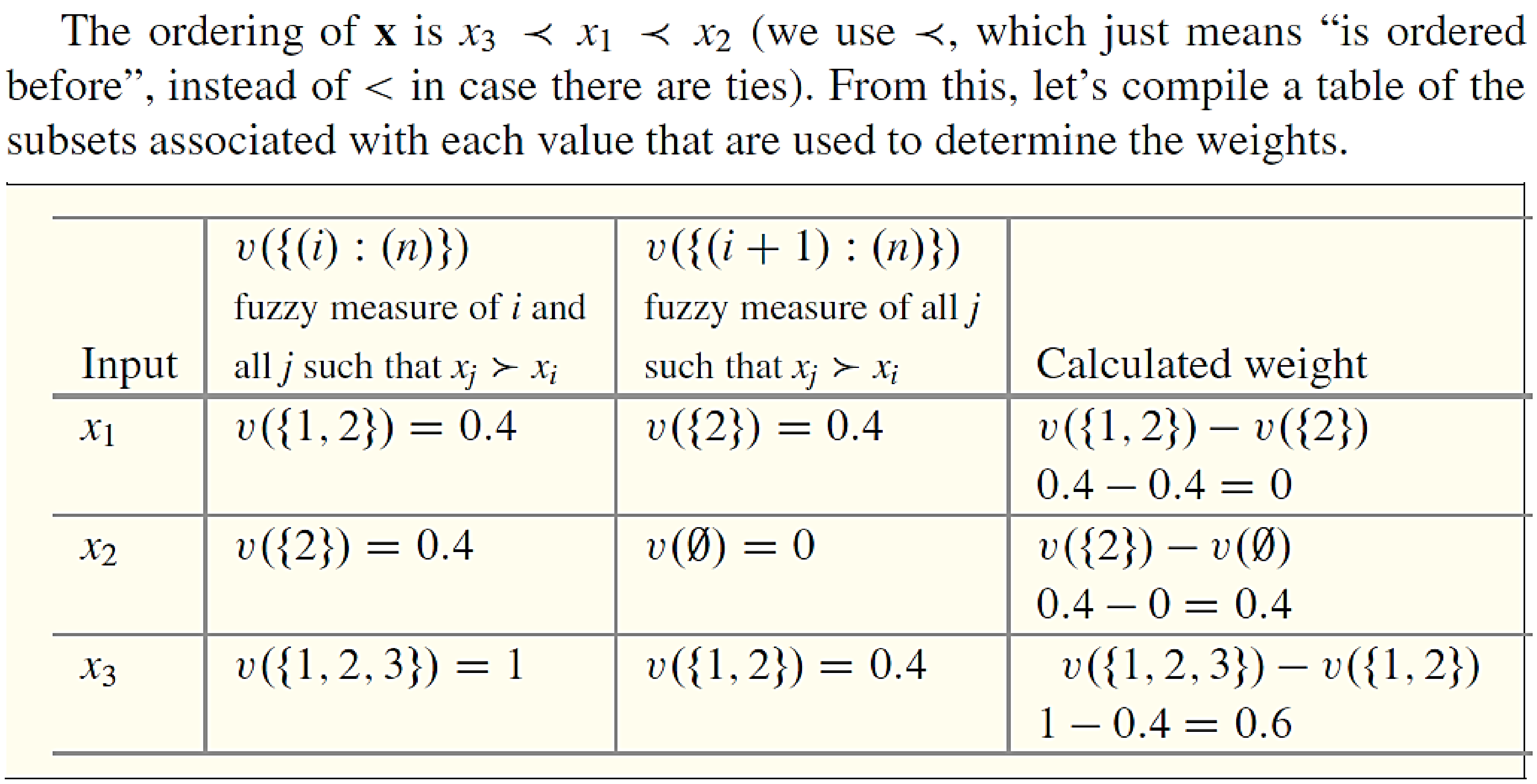
$$v(\{1, 2, 3\}) = 1$$

$$v(\{1, 2\}) = 0.4 \quad v(\{1, 3\}) = 0.9 \quad v(\{2, 3\}) = 0.5$$

$$v(\{1\}) = 0.3 \qquad v(\{2\}) = 0.4 \qquad v(\{3\}) = 0.1$$

$$v(\emptyset) = 0$$

Choquet Integral



The ordering of $\mathbf{x}$ is $x_3 \prec x_1 \prec x_2$ (we use $\prec$, which just means "is ordered before", instead of $<$ in case there are ties). From this, let's compile a table of the subsets associated with each value that are used to determine the weights.

Input	$v(\{(i):(n)\})$ fuzzy measure of i and all j such that $x_j \succ x_i$	$v(\{(i+1):(n)\})$ fuzzy measure of all j such that $x_j \succ x_i$	Calculated weight
x_1	$v(\{1,2\}) = 0.4$	$v(\{2\}) = 0.4$	$v(\{1,2\}) - v(\{2\})$ $0.4 - 0.4 = 0$
x_2	$v(\{2\}) = 0.4$	$v(\emptyset) = 0$	$v(\{2\}) - v(\emptyset)$ $0.4 - 0 = 0.4$
x_3	$v(\{1,2,3\}) = 1$	$v(\{1,2\}) = 0.4$	$v(\{1,2,3\}) - v(\{1,2\})$ $1 - 0.4 = 0.6$

Choquet Integral

```
Cho.integral=function(x,v){
  n = length(x)
  w = array(0,n)
  for(i in 1:(n-1)) {
    w[i] <- v[sum(2^(order(x)[i:n]-1))] -
v[sum(2^(order(x)[(i+1):n]-1))]
  }
  w[n] <- 1- sum(w)
  x =sort(x)
  sum(w*x)
}
x <- c(0.8, 0.3, 0.4)
v <- c(0.3, 0.3, 0.6, 0.7, 0.8, 0.7, 1)

Cho.integral(x,v)
```

```
#Choquet Integral
choquetData <- c(0.8, 0.3, 0.4)
choquetFuzzy <- c(0.3, 0.3, 0.6, 0.7, 0.8, 0.7, 1)
# Expected Solution: 0.5

choquetData2 <- c(0.8, 0.3, 0.4)
choquetFuzzy2 <- c(0.4, 0.1, 0.6, 0.1, 0.6, 0.9, 1)
# Expected Solution: 0.52

studentA <- c(18, 16, 10)
studentB <- c(10, 12, 18)
studentC <- c(14, 15, 15)
studentFuzzy <- c(0.45, 0.45, 0.5, 0.3, 0.9, 0.9, 1)
# Expected Solutions: 13.9, 13.6, 14.9
```