I am naive to the 'Orness' concept.

Please go through below link
https://olympus.mygreatlearning.com/courses/136762/modules/items/7917935?pb_id=19501

you are a great professor of statistics and analytics

I am naive student

Please teach me this

Orness measures how “OR-like” an aggregation is: the closer to 1, the more the result behaves like taking a maximum; the closer to 0, the more it behaves like taking a minimum.[1] [2]

Big picture

- In decision aggregation, the Ordered Weighted Averaging (OWA) operator lets you blend between min, average, median, and max by choosing a weight vector over the sorted inputs.[3] [1]
- Orness is a single number in that characterizes the attitude of those weights: optimistic/“any one good is enough” near 1 (max-like), neutral near 0.5 (average/median-like), and pessimistic/“all must be good” near 0 (min-like).[4] [5] [1]

OWA operator definition

- Given inputs $a_1, \ldots, a_n$, sort them descending to get $b_1 \geq b_2 \geq \cdots \geq b_n$, pick nonnegative weights $W = [w_1, \ldots, w_n]$ that sum to 1, and compute $F(a_1, \ldots, a_n) = \sum_{j=1}^{n} w_j b_j$.[1] [3]
- Special cases via weights:
  - Max: $W = [1, 0, \ldots, 0]$.[1]
  - Average: $W = [\frac{1}{n}, \ldots, \frac{1}{n}]$.[1]
  - Min: $W = [0, \ldots, 0, 1]$.[1]

Orness formula and intuition

- Yager’s orness for weights W is $ orness(W)=\frac{1}{n-1}\sum_{j=1}^{n} (n-j)w_j \in $.[6] [4] [1]
- Interpretation:
  - Higher weight on earlier ranks (near b_1) increases orness toward 1 (max-like).[1]

- Higher weight on later ranks (near b_n) decreases orness toward 0 (min-like).[1]

- Reference points: $\mathrm{orness}(\mathrm{max}) = 1$, $\mathrm{orness}(\mathrm{average}) = 0.5$, $\mathrm{orness}(\mathrm{median}) = 0.5$, $\mathrm{orness}(\mathrm{min}) = 0$.[6] [1]

Andness and dispersion

- Andness is the complement: $\mathrm{andness}(W) = 1 - \mathrm{orness}(W)$.[7] [2]
- Dispersion (how spread the weights are) is often measured by entropy $H(W) = -\sum_j w_j \ln w_j$ or quadratic index $\sum_j w_j^2$; it's independent from orness and controls how sharply you focus on a few ranks versus many.[1]

Why orness matters

- It encodes attitude under uncertainty/aggregation:
  - High orness: optimistic aggregation—one strong criterion can drive the decision.[5] [1]
  - Low orness: conservative aggregation—weakness in any criterion can dominate.[5] [1]
- Used to design weights that reflect linguistic quantifiers like "most," "at least one," or "all," enabling interpretable multi-criteria decision making.[8] [1]

Constructing weights for a target orness

- Goal: choose W with a desired orness level and desired dispersion.
- Common approach: generate a weight "shape" (e.g., geometric, linear, quantifier-based) and tune parameters until the orness matches the target, while ensuring weights are nonnegative and sum to 1.[9] [10]
- Constant-orness families keep the same orness as n varies, which is useful when the number of criteria changes.[11]

Simple numeric example

- Suppose $n = 4$, inputs $a = [0.2, 0.5, 0.9, 0.7]$ so sorted $b = [0.9, 0.7, 0.5, 0.2]$.[1]
  - Max-like: $W =$ gives OWA $= 0.9$, orness $= 1$.[4] [1]
  - Average-like: $W = [0.25, 0.25, 0.25, 0.25]$ gives OWA $= 0.575$, orness $= 0.5$.[6] [1]
  - Min-like: $W =$ gives OWA $= 0.2$, orness $= 0$.[4] [1]

Advanced notes

- Orness has axiomatic characterizations and alternative formulations beyond Yager's original index; these frameworks ensure desirable properties like symmetry between orness and andness and consistency under weight reversals.[12] [13] [7]
- Extensions exist for interval-valued inputs and other aggregation families, generalizing the idea of "closeness to OR" beyond classic OWA.[14] [15] [4]

Quick practice

- Compute orness for $n = 5$ and $W = [0.6, 0.2, 0.1, 0.05, 0.05]$:
  - $\text{orness} = \frac{1}{4}\big(4 \cdot 0.6 + 3 \cdot 0.2 + 2 \cdot 0.1 + 1 \cdot 0.05 + 0 \cdot 0.05\big) = \frac{1}{4}(2.4 + 0.6 + 0.2$ → quite OR-like.[1]

If helpful, the next step can be a guided exercise: pick a target orness (e.g., 0.7) and build a weight vector for your own decision problem, then verify the orness with the formula.[10] [1]

⁂

yes please

Great—let's learn Orness step by step with clear intuition, the key formula, and hands-on practice you can do on any small dataset.

Orness is a number between 0 and 1 that tells how "OR-like" an Ordered Weighted Averaging (OWA) aggregation is: 1 acts like a maximum, 0 acts like a minimum, and around 0.5 acts like an average/median attitude.[21] [22]

What OWA does

- Take n scores, sort them from largest to smallest, then average them with a chosen weight vector W that sums to 1; changing W moves you smoothly between min, mean, and max.[23] [22]
- Formally, if the sorted values are $b_1 \geq b_2 \geq \cdots \geq b_n$ and weights are $W = (w_1, \ldots, w_n)$, OWA output is $ \sum_{j=1}^n w_j b_j $ with all $w_j \geq 0$ and $\sum w_j = 1$.[24] [22]

Orness formula

- For weights $W = (w_1, \ldots, w_n)$ aligned with ranks $b_1, \ldots, b_n$, Yager's orness is $ orness(W)=\frac{1}{n-1}\sum_{j=1}^{n}(n-j)w_j \in $.[22] [25] [23]
- Checks: max weights $W = [1, 0, \ldots, 0]$ give orness $= 1$; equal weights give orness $= 0.5$; min weights $W = [0, \ldots, 0, 1]$ give orness $= 0$.[21]

Intuition you can feel

- Putting more weight near the top rank b_1 makes the result optimistic: a single high score can carry the decision (high orness).[22]
- Putting more weight near the bottom rank b_n makes it cautious: any weak score can drag the result down (low orness).[22]

Mini example

- Scores: [0.2, 0.5, 0.9, 0.7] → sorted $b = [0.9, 0.7, 0.5, 0.2]$.[22]
  - Max-like $W =$: OWA=0.9, orness=1.[25] [21] [22]
  - Mean-like $W = [0.25, 0.25, 0.25, 0.25]$: OWA=0.575, orness=0.5.[23] [22]
  - Min-like $W =$: OWA=0.2, orness=0.[25] [21] [22]

Designing weights by target orness

- Goal: pick W with a desired attitude (orness α) and a desired "spread" (how peaky vs flat).[26] [27]
- A simple approach is to choose a parametric shape (e.g., geometric/linear or quantifier-based) and tune its parameter until the orness equals α, keeping weights nonnegative and summing to 1.[28] [29]
- Quantifier-guided OWA uses linguistic terms like "at least half" or "most" to generate W systematically, giving intuitive control over attitude.[29] [28]

Quick practice 1: compute orness

- For n=5, $W = [0.6, 0.2, 0.1, 0.05, 0.05]$:
  $\text{orness} = \frac{1}{4}(4 \cdot 0.6 + 3 \cdot 0.2 + 2 \cdot 0.1 + 1 \cdot 0.05 + 0 \cdot 0.05) = 0.8125$ → strongly OR-like.[21] [22]

Quick practice 2: build W for α=0.7

- Start with a simple decreasing shape, e.g., $w_j \propto r^{j-1}$ for some r in (0,1), then normalize to sum to 1; adjust r until orness is ~0.7.[27] [26]
- Alternative: use a linguistic-quantifier generator Q to create W that corresponds to "most" for stronger optimism when n is fixed.[28] [29]

Why orness is useful

- It captures the decision attitude in multi-criteria aggregation: higher α rewards any very good criterion; lower α penalizes any very bad one.[21] [22]
- It provides a single, interpretable knob to tune OWA behavior while leaving room to control dispersion (how concentrated the weights are).[23] [22]

Extra pointers

- Andness is $1-$orness; it reflects "AND-like" behavior complementing OR-like behavior.[21]
- There are axiomatic frameworks and extensions of orness for broader aggregation families and interval-valued inputs, useful in advanced settings.[30] [24]

Would you like a guided exercise where you share your own scores and a desired attitude (e.g., "most" or α=0.7), and then we build and verify W together?[29] [28]

⁂

yes, I Would like to have a guided exercise where you share your own scores and a desired attitude (e.g., “most” or α=0.7), and then we build and verify W together

Great choice. Here’s a guided, hands-on exercise building an OWA, computing its orness, and checking how the aggregation changes with the decision attitude.

Orness controls how “OR-like” your aggregation is: nearer 1 behaves like a maximum, nearer 0 behaves like a minimum, and around 0.5 behaves like an average/median.[42] [43]

Step 1: set scores and sort

- Suppose there are 5 criteria with raw scores a = [0.20, 0.55, 0.80, 0.40, 0.95]. Sort them descending to get b = [0.95, 0.80, 0.55, 0.40, 0.20].[42]

Step 2: pick target attitude

- Choose desired orness α = 0.70 to encode an optimistic stance (closer to max but not extreme).[43] [42]

Step 3: build a weight shape

- Use a simple geometric, top-heavy shape: start with unnormalized weights vj = r^(j−1) for j=1..n with r in (0,1); smaller r pushes mass to the top ranks, bigger r spreads it out. Normalize to W by wj = vj / ∑vj.[44] [45]

Step 4: tune r to hit α

- For n=5 and r=0.6, the unnormalized v = [1, 0.6, 0.36, 0.216, 0.1296]; normalize to get W ≈ [0.436, 0.262, 0.157, 0.094, 0.056].[44]
- Compute orness with Yager’s formula $ orness(W)=\frac{1}{n-1}\sum_{j=1}^{n}(n-j)w_j $. For this W: orness ≈ (1/4)(4·0.436 + 3·0.262 + 2·0.157 + 1·0.094 + 0·0.056) ≈ 0.70, which matches the target.[46] [42]

Step 5: compute the OWA value

- OWA = ∑ wj bj = 0.436·0.95 + 0.262·0.80 + 0.157·0.55 + 0.094·0.40 + 0.056·0.20 ≈ 0.79. This is higher than the simple mean due to the optimistic weights.[46] [42]

Step 6: compare against baseline attitudes

- Max-like (α=1): W = gives OWA = 0.95.[47] [43] [42]
- Mean-like (α≈0.5): W = [0.2,0.2,0.2,0.2,0.2] gives OWA = average = 0.58.[46] [42]
- Min-like (α=0): W = gives OWA = 0.20.[43] [47] [42]

Step 7: alternative "linguistic" construction

- Linguistic quantifiers generate W via Q, e.g., Q(r) = r^β for r in; then wi = Q(i/n) − Q((i−1)/n). Larger β concentrates weight toward higher ranks ("most"), yielding higher orness; smaller β is more cautious.[48] [49] [47]
- This approach directly ties W to verbal attitudes like "at least half," "many," or "most," and is commonly used in MCDM.[49] [48]

What you learned and next step

- You set scores, chose a target orness α, built W to match α, and saw how OWA shifts as attitude changes. This gives a practical, tunable way to move between min/mean/max using one parameter.[42] [46]
- Share any 4–8 scores and a target α (or a term like "most"), and a preferred generator (geometric or quantifier). Then a custom W will be produced, verified for orness, and the OWA value computed for your data.[48] [44]

⁂

get type of data in'R'

In R, "type of data" usually means two related things: basic data types (how values are stored) and object structures (how values are organized). To check types, use class(), typeof(), and is.*() helpers.[66] [67] [68]

Basic data types

- numeric (double-precision numbers like 3.14; integers are 3L)[67] [66]
- integer (whole numbers with L suffix, e.g., 10L)[66] [67]
- logical (TRUE, FALSE)[67] [66]
- character (text strings like "hello")[66] [67]
- complex (e.g., 1+2i)[67] [66]
- raw (bytes; less common in day-to-day analysis)[69] [67]

Object structures

- vector: 1D homogeneous collection (all elements same basic type)[70] [71]
- matrix/array: multi-dimensional homogeneous collection[71] [70]
- data.frame: 2D table with columns possibly of different types[70] [71]
- list: ordered collection that can mix types/structures[71] [70]
- factor: categorical vector with levels[68] [70]

How to check type

- class(x): high-level class (e.g., "numeric", "data.frame")[72] [68]
- typeof(x): internal storage (e.g., "double", "integer", "list")[68] [67]
- is.numeric(x), is.character(x), is.factor(x), etc.: TRUE/FALSE checks[72] [68]

Quick examples

- class(10.5) → "numeric"; typeof(10.5) → "double"[68] [66]
- class(100L) → "integer"; typeof(100L) → "integer"[66] [68]
- class(TRUE) → "logical"; is.logical(TRUE) → TRUE[72] [67]
- class("R") → "character"; is.character("R") → TRUE[68] [66]
- class(1+2i) → "complex"; is.complex(1+2i) → TRUE[67] [66]

Data structures in practice

- Vectors: a ← c(1,2,3); class(a) → "numeric"[70] [71]
- Matrices: m ← matrix(1:6, nrow=2); class(m) → "matrix"[71] [70]
- Data frames: df ← data.frame(x=1:3, y=c("a","b","c")); class(df) → "data.frame"[70] [71]
- Lists: L ← list(1, "a", TRUE); class(L) → "list"[71] [70]
- Factors: f ← factor(c("low","high")); class(f) → "factor"[68] [70]

Tips

- Numeric without L is usually stored as double; use as.integer() or add L to ensure integer.[66] [68]
- Use str(x) to see structure and types of nested objects at a glance.[70] [68]
- For categorical data, prefer factor() to capture levels explicitly.[68] [70]

If you share a specific R object (or a small snippet), the exact type and best practices for converting or checking it can be shown step by step.[70] [68]

⁂

class(my.data)

[1] "matrix" "array"
below is the head(my.data). Please change headers to X1,...,X6,Y

```
     V1        V2        V3     V4       V5        V6   V7
```

[1,] 14431 20.70000 33.83333 22.89 32.9000 764.7000 50
[2,] 1694 16.20000 43.96667 15.19 43.2600 769.9333 130
[3,] 6734 19.10000 38.16333 20.29 35.7900 751.0000 130
[4,] 15534 20.85667 33.33333 20.89 34.5000 756.3750 50
[5,] 15675 19.53333 39.02667 19.50 37.1725 755.2000 50
[6,] 6339 19.73000 37.12667 20.29 36.5900 756.0667 90

You can rename matrix column headers in R with colnames(). Since my.data is a matrix/array, do this:

- Set all seven names, mapping V1..V7 to X1..X6,Y:
  - colnames(my.data) ← c("X1","X2","X3","X4","X5","X6","Y")[75] [76]
- If you prefer using the dimnames attribute (equivalent for matrices):
  - dimnames(my.data)[] ← c("X1","X2","X3","X4","X5","X6","Y")[76] [77]
- To check after renaming:
  - head(my.data) and colnames(my.data)[78] [76]

Notes:

- colnames() works for both matrices and data frames; for matrices it sets the column names held in the dimnames attribute.[76]
- You must supply exactly as many names as there are columns (7 here).[78]

Example putting it together:

- colnames(my.data) ← c("X1","X2","X3","X4","X5","X6","Y")
- head(my.data)

This will produce a header line X1 ... X6 Y instead of V1 ... V7 on printing.[76] [78]

⁂

Create 6 histograms for each X variable and Y

Perfect. Here's a clear plan to create six histograms for each of X1 through X6 and Y, using your existing my.data (a matrix) with headers renamed to X1...X6 and Y.

Assumptions:

- Your current data matrix has 7 columns corresponding to X1, X2, X3, X4, X5, X6, and Y, in that order.
- You want one histogram per column, arranged in a 2-row layout (3+3 plus one extra) or any layout you prefer.

Option A: Base R histograms (simple and quick)

- Rename headers (if not already done): colnames(my.data) ← c("X1","X2","X3","X4","X5","X6","Y")
- Plot six histograms in a 2×3 grid:
  - par(mfrow=c(2,3)) # 2 rows, 3 columns
  - for (k in paste0("X", 1:6)) { hist(my.data[, k], main = k, xlab = k) }
  - hist for Y as well: hist(my.data[, "Y"], main = "Y", xlab = "Y")
- Note: This will run all seven histograms; if you truly want six plus Y separate, you can use par(mfrow=c(2,4)) and leave one panel empty or exclude Y.

Option B: ggplot2 with a long format (recommended for nicer visuals and easy arrangement)

- Convert to data frame and reshape to long format:
  - df ← as.data.frame(my.data)
  - colnames(df) ← c("X1","X2","X3","X4","X5","X6","Y")
  - library(tidyr); library(dplyr); library(ggplot2)
  - df_long ← df %>%
    pivot_longer(cols = starts_with("X") | "Y",
    names_to = "Variable", values_to = "Value")
- Then plot histograms by facet:
  - ggplot(df_long, aes(x = Value)) +
    geom_histogram(binwidth = NULL, color = "black", fill = "steelblue") +
    facet_wrap(~Variable, scales = "free_x") +
    labs(title = "Histograms by Variable", x = "Value", y = "Frequency") +
    theme_minimal()
- If you want separate panels but fixed x scales, you can set scales = "fixed" or define binwidths.

Option C: Six separate ggplots arranged with patchwork (for precise control)

- After creating df_long as above, you can alternatively create six plots and arrange them:
  - library(ggplot2); library(patchwork)
  - pX1 ← ggplot(subset(df_long, Variable == "X1"), aes(Value)) +
    geom_histogram(binwidth = 5) + ggtitle("X1")
  - Repeat for X2–X6 and optionally Y
  - (pX1 | pX2 | pX3) / (pX4 | pX5 | pX6) would create a 2×3 grid
- Adjust binwidths to suit your data distribution.

Practical steps to execute (copy-paste-ready snippets)

- Base R (all seven histograms in a 2×4 grid; Y included)
  - colnames(my.data) ← c("X1","X2","X3","X4","X5","X6","Y")
  - par(mfrow=c(2,4), mar=c(4,4,2,1))
  - for (i in 1:7) {
    hist(my.data[, i], main = colnames(my.data)[i], xlab = colnames(my.data)[i])
    }
- Base R (six X variables only, Y on its own panel)
  - colnames(my.data) ← c("X1","X2","X3","X4","X5","X6","Y")
  - par(mfrow=c(2,3))
  - for (i in 1:6) {
    hist(my.data[, i], main = paste0("X", i), xlab = paste0("X", i))
    }
  - hist(my.data[, 7], main = "Y", xlab = "Y")
- ggplot2 (long format, all in facets)
  - library(tidyr)
  - df ← as.data.frame(my.data)
  - colnames(df) ← c("X1","X2","X3","X4","X5","X6","Y")
  - df_long ← df %>% pivot_longer(cols = c(starts_with("X"), "Y"), names_to = "Variable", values_to = "Value")
  - library(ggplot2)
  - ggplot(df_long, aes(x = Value)) + geom_histogram(binwidth = (diff(range(df_long$Value))/30), fill = "steelblue", color = "black") + facet_wrap(~Variable, scales = "free_x") + theme_minimal()

Notes and tips

- If you prefer a fixed number of bins across histograms, set binwidth or breaks explicitly, e.g., binwidth = 0.5 or breaks = seq(min, max, by = 1).
- If your data contain missing values, add na.rm = TRUE to hist(), i.e., hist(my.data[, i], na.rm = TRUE, ...). Base R's hist() does not have na.rm, so you should do data ← na.omit(my.data[, i]) before plotting.
- For large datasets, consider histogram efficiency and bin width selection to avoid overplotting or under-representation.

If you share a snippet of your exact R environment (base R vs. tidyverse) or your preferred visualization style, the exact code can be tailored and tested step-by-step.

⁂

I ← c("define your variable index") # Choose any four X variables (from X1, X2, X3, X4, X5) and Y

Here's how to define I so it selects any four X variables (from X1–X5) plus Y from your matrix with column names X1,...,X6,Y.

- By column indices (simple and robust):
  - I ← c(1, 2, 4, 5, 7) # e.g., choose X1, X2, X4, X5, and Y [106] [107]
  - my.data[, I] # subset those columns [108] [106]
- By column names (clear and safe if names are set):
  - I ← c("X1", "X3", "X4", "X5", "Y") # pick four Xs and Y [109] [108]
  - my.data[, I] # subset by names [109] [108]
- Programmatically "any four Xs" plus Y (random example):
  - set.seed(1) # for reproducibility
  - Xcols ← paste0("X", 1:5)
  - I ← c(sample(Xcols, 4), "Y")
  - my.data[, I] [108]
- If you prefer fixed choices, here are a few ready-to-use examples:
  - I ← c("X1","X2","X3","X4","Y") [109] [108]
  - I ← c("X1","X2","X4","X5","Y") [108] [109]
  - I ← c(2,3,4,5,7) # X2,X3,X4,X5,Y by index [107] [106]

After defining I, you can proceed with histograms only for those columns:

- par(mfrow=c(2,3)); for (k in I) hist(my.data[, k], main = k, xlab = k) [110] [111]

⁂

Weighted Quasi-Arithmetic: please illustrate

Weighted quasi-arithmetic means generalize many familiar averages by "transforming → averaging → back-transforming." With a strictly monotone generator function g and nonnegative weights w1,...,wn summing to 1, the weighted quasi-arithmetic mean of numbers x1,...,xn is:

- Qg,w(x1,...,xn) = g^{-1}(w1 g(x1) + ... + wn g(xn)), where g^{-1} is the inverse of g. [117]

Why this is useful

- Different choices of g recover standard means: arithmetic, geometric, harmonic, power means, and more, all within one formula. [117]
- Weights w let each observation contribute according to importance or reliability. [118]

Classic special cases via g

- Arithmetic mean: g(x) = x ⇒ Q = ∑ wi xi.[117]
- Geometric mean: g(x) = ln x (x>0) ⇒ Q = exp(∑ wi ln xi) = ∏ xi^{wi}.[117]
- Harmonic mean: g(x) = 1/x (x>0) ⇒ Q = 1 / (∑ wi / xi).[117]
- Power mean of order p: g(x) = x^p (x≥0), p≠0 ⇒ Q = (∑ wi x_i^p)^{1/p}; as p→0, this tends to the geometric mean.[117]

Numeric illustration (weighted)

Suppose x = (2, 5, 8) with weights w = (0.2, 0.3, 0.5).

- Arithmetic (g(x)=x): Q = 0.2·2 + 0.3·5 + 0.5·8 = 6.1.[117]
- Geometric (g(x)=ln x): Q = exp(0.2 ln2 + 0.3 ln5 + 0.5 ln8) ≈ exp(0.1386 + 0.4828 + 1.0397) ≈ exp(1.6611) ≈ 5.27.[117]
- Harmonic (g(x)=1/x): Q = 1 / (0.2/2 + 0.3/5 + 0.5/8) = 1 / (0.1 + 0.06 + 0.0625) = 1 / 0.2225 ≈ 4.49.[117]
- Power mean p=2 (quadratic): Q = (0.2·2^2 + 0.3·5^2 + 0.5·8^2)^{1/2} = (0.8 + 7.5 + 32)^{1/2} = (40.3)^{1/2} ≈ 6.35.[117]

These demonstrate how changing g changes the “emphasis”:

- Harmonic < Geometric < Arithmetic < Quadratic when values vary (a standard ordering).[117]

Properties to remember

- Monotonicity: If each xi ≤ yi then Qg,w(x) ≤ Qg,w(y).[117]
- Idempotence: If all xi=x, then Qg,w = x (any weights), so it’s a true mean.[117]
- Weighting: Larger wi pulls the mean toward xi in g-space (before back-transforming).[118]

How to use in practice

- Choose g to reflect the trade-off attitude:
  - g(x)=x for linear averaging.[117]
  - g concave (e.g., ln x) reduces the influence of large values (risk-averse, multiplicative effects).[117]
  - g convex (e.g., x^2) emphasizes larger values (risk-seeking or penalizing large deviations).[117]
- Choose weights w to reflect importance or reliability; they must sum to 1.[118]

Quick R snippet

- Arithmetic (g(x)=x): sum(w*x)
- Geometric (g(x)=log): exp(sum(w*log(x)))
- Harmonic (g(x)=1/x): 1/sum(w/x)
- Power mean p: (sum(w*x^p))^(1/p)
  These lines compute the corresponding weighted quasi-arithmetic means directly.[117]

If you share your actual vector and chosen g (or a target behavior like “multiplicative” → use ln), a customized calculation and visualization can be provided.[118] [117]

⁂

Explain a simple example of a weighted quasi arithmetic mean

A weighted quasi-arithmetic mean transforms the data, takes a weighted average in the transformed space, then back-transforms the result. Formally, with a strictly monotone generator g and weights w1,...,wn summing to 1, it is Qg,w(x1,...,xn) = g^{-1}(∑ wi g(xi)).[125]

Intuition

- Pick a transformation g that encodes how you want to emphasize values, average in g-space using the weights, then invert g to return to the original scale. This single template recovers many familiar means by changing g, while weights control each value’s influence.[126] [125]

Simple numeric example

- Data: x = (2, 5, 8), weights w = (0.2, 0.3, 0.5), which sum to 1.[126]
- Arithmetic mean (g(x)=x): Q = 0.2·2 + 0.3·5 + 0.5·8 = 6.1, since g^{-1} is identity.[125]
- Geometric mean (g(x)=ln x, x>0): Q = exp(0.2 ln 2 + 0.3 ln 5 + 0.5 ln 8) ≈ exp(1.6611) ≈ 5.27.[125]
- Harmonic mean (g(x)=1/x, x>0): Q = 1 / (0.2/2 + 0.3/5 + 0.5/8) = 1 / 0.2225 ≈ 4.49.[125]

What this shows

- Changing g changes the “attitude” of the mean: harmonic < geometric < arithmetic for varying positive data, reflecting increasing emphasis on larger values. This ordering is a standard behavior of these choices of g.[125]
- Weights tilt the mean toward more important observations in the transformed space g(x), before mapping back with g^{-1}.[126]

One more variant: power mean

- Choose g(x)=x^p (p≠0) to get the weighted power mean: Q = (∑ wi x_i^p){1/p}; as p→0 it approaches the geometric mean, connecting these families smoothly.[125]

Quick R snippets

- Arithmetic (g(x)=x): sum(w*x).[125]
- Geometric (g(x)=log): exp(sum(w*log(x))).[125]
- Harmonic (g(x)=1/x): 1/sum(w/x).[125]

If you share your x and the behavior you prefer (e.g., multiplicative effects → use ln; penalize large values less → concave g), a tailored weighted quasi-arithmetic mean can be computed and compared to alternatives.[126] [125]

⁂

what is this 'R' expression

hotelData[,c(6,5,10,91)]

It subsets columns 6, 5, 10, and 91 (in that order) from the object hotelData. In base R, x[rows, cols] uses the part before the comma to pick rows and the part after the comma to pick columns; leaving rows blank means “all rows,” and c(6,5,10,91) is the vector of column indices to keep, reordered as specified. If hotelData is a data frame or matrix with at least 91 columns, the result is a new object containing only those four columns; if any of those indices exceed the number of columns, R will throw an error.

⁂

how about hotelData[,c(2:10,9)]

It selects columns 2 through 10, and then repeats column 9 at the end, from all rows of hotelData. In base R, x[rows, cols] subsets; leaving rows blank means “all rows,” and c(2:10, 9) builds the column index vector 2,3,4,5,6,7,8,9,10,9. If hotelData has at least 10 columns, the result will have 10 columns where the 9th column appears twice (once in position 8 and again as the last column); duplicated columns are allowed when subsetting by indices.

⁂

<q>Option B: ggplot2 with a long format (recommended for nicer visuals and easy arrangement)

Convert to data frame and reshape to long format:

df ← as.data.frame(my.data)

colnames(df) ← c("X1","X2","X3","X4","X5","X6","Y")

library(tidyr); library(dplyr); library(ggplot2)

df_long ← df %>%
pivot_longer(cols = starts_with("X") | "Y",
names_to = "Variable", values_to = "Value")

Then plot histograms by facet:

ggplot(df_long, aes(x = Value)) +
geom_histogram(binwidth = NULL, color = "black", fill = "steelblue") +
facet_wrap(~Variable, scales = "free_x") +
labs(title = "Histograms by Variable", x = "Value", y = "Frequency") +
theme_minimal()</q> Please share complete code

Here's a complete, copy-paste-ready script that renames your matrix columns to X1–X6 and Y, reshapes to long format, and draws faceted histograms with sensible defaults (automatic binning and free x-scales).

- Assumes my.data is a 7-column matrix (or data frame).[152] [153]

Code:

- # Rename columns
- df ← as.data.frame(my.data)
- colnames(df) ← c("X1","X2","X3","X4","X5","X6","Y")
-
- # Load packages
- library(tidyr)
- library(dplyr)
- library(ggplot2)
-

- # Long format for ggplot facets
- df_long ← df %>%
- tidyr::pivot_longer(
- ```
  cols = c(dplyr::starts_with("X"), "Y"),
  ```
- ```
  names_to = "Variable",
  ```
- ```
  values_to = "Value"
  ```
-)
-
- # Faceted histograms
- ggplot(df_long, aes(x = Value)) +
- geom_histogram(color = "black", fill = "steelblue") +
- facet_wrap(~ Variable, scales = "free_x") +
- labs(title = "Histograms by Variable", x = "Value", y = "Frequency") +
- theme_minimal()

Notes and options:

- The pivot_longer with c(starts_with("X"), "Y") selects X1–X6 and Y explicitly; adjust if your column names differ.[154] [152]
- geom_histogram() without binwidth lets ggplot2 choose bins via the Freedman–Diaconis or Sturges heuristics; set binwidth = value or bins = n to control it.[155] [152]
- facet_wrap(~ Variable, scales = "free_x") uses separate x-scales per variable, which is helpful when variables are on different ranges.[156] [152]
- If you see warnings about NAs, filter them out with df_long ← df_long %>% filter(!is.na (Value)).[153] [157]

⁂
  ```

Context and Reference : We highly recommend looking through the attached reference for data analysis case study. From ‘Chapter 5.4 Using Aggregation Models for Analysis and Prediction’ onwards, we investigate Bike Sharing data (https://olympus.mygreatlearning.com/courses/136762/files/12809029/download?verifier=t0Zpvf30GLeqWdjpGyUh2lfHRGBBuulZk54yOtJO&wrap=1), and there is significantly more exploration and decisions made in terms of determining appropriate transformations.

'AggWaFit718' is R file with fitting models which perform 'Weight Analysis' to yield optimum models. (you need to install.packages("lpSolve"))

The fitting functions in AggWaFit718.r takes in data as its input where it treats the last column as our intended output values, and the other columns as input values. It then attempts to find weights for each input by minimising the error measure, which requires optimisation knowledge (not assessed) of which the basics are covered in these weeks
Details for this can be found in Chapter 5.3 of the prescribed textbook.
The aim here is to now run these fitting functions for various models so that we can collect data on the errors, correlation, and weights. i.e. Weighted Arithmetic Mean , power mean (p.=0.5),Quadratic mean (p=2), ordered weighted average, choquet weight (shapley values) for fuzzy measures

Please refer this for conversation

The attached R file and textbook chapter together define a framework for *fitting* several aggregation models (WAM, power means, OWA, Choquet) to data so that you can compare errors, correlations, and learned weights.[158] [159]

What AggWaFit718.R does

- The script defines generator functions for several quasi-arithmetic / power means, such as AM, GM, QM, PM05, along with their inverses (g and g.inv).[158]
- It then defines higher-level fitting functions that treat the last column of a data matrix as the output y and the preceding columns as inputs $x_1, \ldots, x_n$, and uses linear programming (lpSolve) to find weights that minimize a least absolute deviation (LAD) error.[159] [158]

Key fitting functions (all expect a numeric matrix with last column as output):

- `fit.QAM(the.data, output.1, stats.1, g, g.inv)`
  Fits a weighted quasi-arithmetic mean (includes weighted arithmetic and power means as special cases).[159] [158]
- `fit.OWA(the.data, output.1, stats.1)`
  Fits an Ordered Weighted Averaging operator (weights attached to ordered inputs).[158] [159]
- `fit.choquet(the.data, output.1, stats.1, k.add)`
  Fits a Choquet integral by learning a fuzzy measure and optionally its Shapley values and an orness index.[159] [158]

Each fitting function writes two files:

1. the original data plus predicted $ \hat{y} $ as a final column, and
2. a "stats" file with RMSE, average absolute error, Pearson and Spearman correlations, and the learned weights (or fuzzy measure / Shapley values).[158] [159]

How this matches Chapter 5 (bike-sharing example)

- Chapter 5 uses these functions to fit models to transformed BikeShare data, always using the last column as the transformed output (e.g., number of casual users) and first columns as transformed predictors (weather, temperature, humidity, windspeed).[159]
- For weighted arithmetic / power / quasi-arithmetic means, the linear program finds weights w_j such that a chosen quasi-arithmetic mean $Q_{g,w}(x)$ is as close as possible to the observed y in LAD sense.[159]
- For OWAs, the data in each row are sorted and weights are attached to order statistics, again using LAD to estimate weights.[158] [159]
- For the Choquet integral, the script uses the Möbius representation of the fuzzy measure plus monotonicity and normalization constraints to fit a measure that best models the data, and then derives Shapley indices and orness.[158] [159]

How you would typically use it in R

Below is a complete "skeleton" for your own analysis (any dataset with inputs in columns 1:(n−1) and output in column n):

```
# 1. Install and load lpSolve once
install.packages("lpSolve")   # only once
library(lpSolve)

# 2. Load the AggWaFit functions (place AggWaFit718.R in your working directory)
source("AggWaFit718.R")

# 3. Load and (optionally) transform your data
# Example: data matrix with inputs in columns 1:(n-1) and output in column n
my.data <- as.matrix(read.table("YourData.txt"))

# 4. Fit weighted arithmetic mean (default g = AM)
fit.QAM(my.data,
        output.1 = "wam_output.txt",
```

```
        stats.1  = "wam_stats.txt")   # WAM model

# 5. Fit specific power means via generators in AggWaFit718.R

# p = 0.5 power mean
fit.QAM(my.data,
        output.1 = "pm05_output.txt",
        stats.1  = "pm05_stats.txt",
        g        = PM05,
        g.inv    = invPM05)

# p = 2 quadratic mean
fit.QAM(my.data,
        output.1 = "qm_output.txt",
        stats.1  = "qm_stats.txt",
        g        = QM,
        g.inv    = invQM)

# Approximate geometric mean (uses GMa / invGMa as done in the book)
fit.QAM(my.data,
        output.1 = "gm_output.txt",
        stats.1  = "gm_stats.txt",
        g        = GMa,
        g.inv    = invGMa)

# 6. Fit OWA
fit.OWA(my.data,
        output.1 = "owa_output.txt",
        stats.1  = "owa_stats.txt")

# 7. Fit Choquet integral (kadd controls k-additivity if you want)
fit.choquet(my.data,
            output.1 = "choquet_output.txt",
            stats.1  = "choquet_stats.txt")
```

In each case, you then:

- Open the corresponding `*_stats.txt` file to read RMSE, average absolute error, Pearson/Spearman correlations, and the learned weights or fuzzy measure/Shapley indices. [159] [158]
- Compare models: lower RMSE / average absolute error and higher correlation indicate better fit; more flexible models (e.g., Choquet) can overfit if data are limited, so Chapter 5 also discusses overfitting concerns and the interpretation of weights.[159]

How this connects to your earlier questions

Given your earlier work on weighted quasi-arithmetic means and Orness:

- The `fit.QAM` calls implement exactly the "weighted quasi-arithmetic" fitting you asked about, where functions like the weighted arithmetic mean, geometric mean, and power means are all special cases obtained via different generators g.[158] [159]

- The `fit.OWA` and `fit.choquet` calls extend your weight analysis to ordered weights and fuzzy measures, providing orness and Shapley indices that help interpret whether the learned models behave more like “AND”, “OR”, or something in between.[158] [159]

If you tell which dataset you plan to start with (BikeShare or your own X1–X6,Y matrix) and which model you want to run first (e.g., WAM vs. OWA), the next step can be a line-by-line walkthrough of that exact fit and how to interpret its stats file.[159] [158]

⁂

KeilHotels is the dataset.

role : data scientist and analyst
task : while performing below task please illustrate by and provide 'R' code.
fit all models and perform 1. Error/Correlation Analysis 2. Weight Analysis 3. Orness Analysis 4. Fuzzy Measures Analysis

Below are detailed instructions further :
Error/Correlation Analysis:
Tabulate your results for both errors and both correlations for all measures. In general we’re looking for low error values and correlations close to 1.
Which of these models appear to the best out of the whole lot we’ve tested?
Which models seem to be relatively similar to each other?
Why do you think that might be?
Weight Analysis:
For each of the models, which variable (and corresponding user) do the weights favour? What does this tell us about their similarity to Kei’s ratings? Are all the models similar in their preference? Which models appear to have slightly different preferences and why do you think that might be the case?
Note that the Shapley values for the Choquet Integral can be considered as ‘weights’ of each individual variable for the purposes of weight analysis.
There is always the possibility that the weights may be extremely skewed towards certain variables than others, which is why it’s important to test a variety of models. The Bike Sharing example referenced in the previous article demonstrates an instance of this happening.
Orness Analysis:
For the OWA and Choquet Integral, what does the orness tell us about their averaging tendencies with respect to this set of inputs and associated weights?
Fuzzy Measures Analysis:
Which combinations of inputs are additive, superadditive or redundant?
Feel free to post your numbers and thoughts in the comments below so that we can ensure that everyone’s results match!

KeiHotels can be treated as a matrix with 9 input columns (users’ ratings) and the 10th column as Kei’s overall rating, and AggWaFit718.R gives you ready-made functions to fit WAM, power means, OWA, and Choquet to this data.[160] [161]

Below is a complete, step-by-step R workflow: load data, fit all models, then extract errors, correlations, weights, orness, and fuzzy measures for your analysis.[161] [162] [160]

1. Load KeiHotels and AggWaFit code

```
# 1. Set working directory to where files are saved
# setwd("path/to/your/folder")

# 2. Read KeiHotels data (tab-separated, no header)
kei <- as.matrix(read.table("KeiHotels.txt", header = FALSE))

dim(kei)      # should be (number_of_hotels, 10)
head(kei)

# 3. Name columns: 9 users + Kei
colnames(kei) <- c(paste0("U", 1:9), "Kei")

# 4. Load lpSolve and AggWaFit functions
install.packages("lpSolve")     # run once if not installed
library(lpSolve)

source("AggWaFit718.R")         # this file defines QAM, OWA, choquet, fit.* etc.
```

The AggWaFit script assumes the last column is the output y and all previous columns are inputs $x_1, \ldots, x_n$.[162] [160]

2. Fit all requested models

You will fit: weighted arithmetic mean, power mean $p = 0.5$, quadratic mean $p = 2$, OWA, and Choquet.[160] [162]

```
dataMat <- kei   # just to shorten the name

## 1) Weighted Arithmetic Mean (WAM) – QAM with g = AM
fit.QAM(
  the.data = dataMat,
  output.1 = "kei_wam_output.txt",
  stats.1  = "kei_wam_stats.txt"   # will contain RMSE, avg abs error, correlations, orne
)

## 2) Power mean with p = 0.5
# PM05 and invPM05 are defined in AggWaFit718.R
fit.QAM(
  the.data = dataMat,
  output.1 = "kei_pm05_output.txt",
  stats.1  = "kei_pm05_stats.txt",
  g        = PM05,
  g.inv    = invPM05
)

## 3) Quadratic mean with p = 2
fit.QAM(
  the.data = dataMat,
  output.1 = "kei_qm_output.txt",
```

```
  stats.1  = "kei_qm_stats.txt",
  g        = QM,
  g.inv    = invQM
)

## 4) Ordered Weighted Average (OWA)
fit.OWA(
  the.data = dataMat,
  output.1 = "kei_owa_output.txt",
  stats.1  = "kei_owa_stats.txt"
)

## 5) Choquet Integral (fuzzy measure + Shapley + orness)
# k.add controls k-additivity; you can start with k.add = 2 (2-additive measure)
fit.choquet(
  the.data = dataMat,
  output.1 = "kei_choquet_output.txt",
  stats.1  = "kei_choquet_stats.txt",
  k.add    = 2
)
```

Each fit writes: (a) an output file with original data plus predicted Kei rating (last column), and (b) a stats file with RMSE, average absolute error, Pearson and Spearman correlations, orness, and (for Choquet) Shapley indices and fuzzy measure values.[162] [160]

3. Error and correlation analysis

The stats files have a standard layout; for Choquet, for example, the first rows are:

- "RMSE", value
- "Av. abs error", value
- "Pearson Correlation", value
- "Spearman Correlation", value
- "Orness", value
  then Shapley and fuzzy measure tables.[160]

You can build a summary table across all models as follows.[162] [160]

```
read_stats <- function(file) {
  s <- read.table(file, stringsAsFactors = FALSE)
  out <- list(
    RMSE      = as.numeric(s[1, 2]),
    L1        = as.numeric(s[2, 2]),
    Pearson   = as.numeric(s[3, 2]),
    Spearman  = as.numeric(s[4, 2]),
    Orness    = as.numeric(s[5, 2])
  )
  out
}

wam   <- read_stats("kei_wam_stats.txt")
```

```
pm05  <- read_stats("kei_pm05_stats.txt")
qm    <- read_stats("kei_qm_stats.txt")
owa   <- read_stats("kei_owa_stats.txt")
chq   <- read_stats("kei_choquet_stats.txt")

error_cor_summary <- data.frame(
  Model    = c("WAM", "PM0.5", "QM2", "OWA", "Choquet"),
  RMSE     = c(wam$RMSE, pm05$RMSE, qm$RMSE, owa$RMSE, chq$RMSE),
  L1       = c(wam$L1, pm05$L1, qm$L1, owa$L1, chq$L1),
  Pearson  = c(wam$Pearson, pm05$Pearson, qm$Pearson, owa$Pearson, chq$Pearson),
  Spearman = c(wam$Spearman, pm05$Spearman, qm$Spearman, owa$Spearman, chq$Spearman),
  Orness   = c(wam$Orness, pm05$Orness, qm$Orness, owa$Orness, chq$Orness)
)
error_cor_summary
```

Interpretation guide: models with lower RMSE and average absolute error and higher Pearson/Spearman correlations (close to 1) fit Kei's ratings better.[161] [162]

Typical patterns to look for:

- The three quasi-arithmetic models (WAM, PM0.5, QM2) often have similar errors and correlations if Kei's ratings are roughly linear in the users' ratings; PM0.5 may behave closer to geometric and can differ when ratings interact multiplicatively.[162]
- OWA is more flexible in how it emphasizes ordered inputs; it may slightly improve error or correlation and reflect "optimistic" or "pessimistic" attitudes.[162]
- Choquet can capture interactions between users (e.g., "both user A and user B being high matters more than either alone"), so it can improve fit but may produce more complex, sometimes skewed measures.[160] [162]

From your `error_cor_summary` you can answer:

- **Best model**: the one with lowest RMSE and L1 plus highest correlations.[162]
- **Similar models**: those with very close RMSE/L1 and correlations (e.g., WAM and PM0.5 might be very similar, indicating the transformation has little effect).[162]

4. Weight analysis

For QAM- and OWA-fits, the stats files also include the learned weights after the error rows.[160]

For Choquet, the Shapley indices in the stats file serve as "overall weights" of each variable.[160] [162]

4.1 Extract weights for WAM / PM0.5 / QM2 / OWA

In each of these stats files, there will be a block labeled something like `"i","weight i"` followed by rows `1, w1`, `2, w2`, ..., one per variable.[160]

You can extract them like this (assuming the same format as in AggWaFit):[160]

```
read_weights <- function(file, n_inputs = 9) {
  s <- read.table(file, stringsAsFactors = FALSE)
  # Find the row where weights start: first row whose first column is "i"
  idx <- which(s[,1] == "i")[^14_1]
  w_block <- s[(idx + 1):(idx + n_inputs), ]
  w <- as.numeric(w_block[, 2])
  names(w) <- paste0("U", 1:n_inputs)
  w
}

w_wam  <- read_weights("kei_wam_stats.txt")
w_pm05 <- read_weights("kei_pm05_stats.txt")
w_qm   <- read_weights("kei_qm_stats.txt")
w_owa  <- read_weights("kei_owa_stats.txt")

w_wam
w_pm05
w_qm
w_owa
```

Interpretation:

- Variables (users) with larger weights are more influential for that model's prediction of Kei's ratings.[161] [162]
- If, for example, user U3 and U7 have high weights in all models, it suggests Kei's ratings align most closely with those users' preferences.[161] [162]
- If some model (e.g., QM2) shows more skewed weights toward extremes, this reflects how its generator $g(x) = x^2$ emphasizes larger inputs.[162] [160]

4.2 Shapley values as "weights" for Choquet

The Choquet stats file contains a block:

- Row: `"i","Shapley i"`
- Then one row per variable `i`, giving the Shapley index.[160]

```
read_shapley <- function(file, n_inputs = 9) {
  s <- read.table(file, stringsAsFactors = FALSE)
  idx <- which(s[,1] == "i" & s[,2] == "Shapley")[^14_1]  # or just s[,1]=="i"
  # next n_inputs rows contain i and shapley(i)
  sh_block <- s[(idx + 1):(idx + n_inputs), ]
  sh <- as.numeric(sh_block[, 2])
  names(sh) <- paste0("U", 1:n_inputs)
  sh
}

sh_chq <- read_shapley("kei_choquet_stats.txt")
sh_chq
```

Interpretation:

- Shapley values represent the average marginal contribution of each user over all coalitions; they can be treated as "global weights" for Choquet.[162] [160]
- Compare Shapley values with WAM/OWA weights: if they mostly agree on which users are important, the models are consistent; if Choquet highlights different users, it may be capturing interaction effects that simple means miss.[161] [160] [162]

From these, answer:

- Which users are most favored in each model.[161] [162]
- Whether all models share similar preferences (similar weight rankings) or if, say, OWA or Choquet shift importance because of nonlinearity and interactions.[160] [162]

5. Orness analysis

The stats files already include an "Orness" value for each model; for Choquet, this is computed from the fuzzy measure via `orness.v`, and for OWA from the OWA weights.[160]

You already loaded them into `error_cor_summary$Orness`:

```
error_cor_summary[, c("Model", "Orness")]
```

Interpretation:

- Orness near 1: model behaves more like an OR (similar to max) – high ratings by *some* users are enough to push Kei's rating up.[162] [160]
- Orness near 0: model behaves more like an AND (similar to min) – all (or most) users must rate highly for Kei to rate highly.[162] [160]
- Orness near 0.5: more "compensatory" averaging, closer to simple mean or median behavior.[160] [162]

Answer for your data:

- For **OWA**, comment on whether Kei's rating tends to follow the best ratings (high orness), the worst ratings (low orness), or a balanced mix (middle orness).[161] [162] [160]
- For **Choquet**, the orness reflects the underlying fuzzy measure and tells you whether coalitions of high raters are treated more optimistically or conservatively.[162] [160]

6. Fuzzy measures analysis (additive, superadditive, redundant)

The Choquet stats file also prints the fuzzy measure values $m(A)$ for all non-empty subsets A of the 9 users, encoded by a "binary number" column.[160]

From the snippet in AggWaFit, the block looks like:

- Row: `"binary number","fm.weights"`

- Then rows `k, m(A_k)` for k = 1,...,2^n − 1, where k's binary representation encodes which users are in subset A_k.[160]

You can read these and test additivity:

- For singletons i: index is 2^{i-1}.
- For pairs i, j: index is $2^{i-1} + 2^{j-1}$.[160]

```
read_fm <- function(file, n_inputs = 9) {
  s <- read.table(file, stringsAsFactors = FALSE)
  idx <- which(s[,1] == "binary" & s[,2] == "number")[^14_1]  # row with header
  fm_block <- s[(idx + 1):nrow(s), ]
  # first column is index k, second column is fuzzy measure value
  k  <- as.integer(fm_block[, 1])
  fm <- as.numeric(fm_block[, 2])
  out <- numeric(2^n_inputs)
  out[k + 1] <- fm   # fm[2:(2^n)] in AggWaFit
  out
}

fm <- read_fm("kei_choquet_stats.txt")  # fm[^14_0] unused, fm[2^n] = 1 (full set)

# Helper to get m(A) for a set of indices e.g. c(1,3) for U1 and U3
m_set <- function(vars) {
  idx <- sum(2^(vars - 1))
  fm[idx + 1]
}

# Check pairwise relationships for all pairs
n_inputs <- 9
pair_analysis <- data.frame()
for (i in 1:(n_inputs - 1)) {
  for (j in (i + 1):n_inputs) {
    m_i  <- m_set(i)
    m_j  <- m_set(j)
    m_ij <- m_set(c(i, j))
    rel <- if (abs(m_ij - (m_i + m_j)) < 1e-6) {
      "additive"
    } else if (m_ij > m_i + m_j) {
      "superadditive"
    } else {
      "subadditive (redundant)"
    }
    pair_analysis <- rbind(pair_analysis,
                           data.frame(Ui = i, Uj = j,
                                      m_i = m_i, m_j = m_j,
                                      m_ij = m_ij, relation = rel))
  }
}
pair_analysis
```

Interpretation:

- **Additive**: $m(\{i,j\}) = m(\{i\}) + m(\{j\})$ → users i and j contribute independently; Choquet reduces to a weighted sum for these.[162] [160]
- **Superadditive**: $m(\{i,j\}) > m(\{i\}) + m(\{j\})$ → synergy; having both high is worth more than the sum of individual contributions.[162] [160]
- **Subadditive / redundant**: $m(\{i,j\}) < m(\{i\}) + m(\{j\})$ → redundancy; they overlap in what they capture about Kei's preferences.[162] [160]

You can summarize:

- Which pairs (Ui, Uj) are superadditive (synergistic combinations).[161] [160]
- Which are strongly redundant (subadditive with big gap), suggesting that those users have very similar rating behavior from Kei's perspective.[160] [162]

7. How to write up your answers

Using the outputs you compute:

1. **Error/Correlation analysis**:
   - Present the `error_cor_summary` table, highlight the smallest RMSE/L1 and largest correlations, and declare the best model.[161] [162]
   - Note which models are numerically close, e.g., WAM vs PM0.5, and connect that to the idea that small changes in generator g do not dramatically change fit for this dataset.[162]
2. **Weight analysis**:
   - For each model, list the top 2–3 users by weight (or Shapley for Choquet) and comment on how aligned they are with Kei's ratings.[160] [162]
   - Mention any model that shows a clearly different preference pattern (e.g., OWA emphasizing "best of" users, Choquet emphasizing synergistic pairs).[162] [160]
3. **Orness analysis**:
   - Use the Orness column to describe whether OWA and Choquet behave more "optimistically" or "pessimistically" in aggregating the users' ratings.[160] [162]
4. **Fuzzy measures analysis**:
   - From `pair_analysis`, list a few additive, superadditive, and redundant pairs, and interpret them as independent, synergistic, or overlapping user influences.[161] [162] [160]

If you paste your actual `error_cor_summary`, one set of weight vectors, and a few rows from `pair_analysis`, a detailed interpretation tailored to your KeiHotels results can be provided.[161] [162] [160]

⁂

<q>fit.OWA(

the.data = dataMat,
output.1 = "kei_owa_output.txt",
stats.1 = "kei_owa_stats.txt"
)</q> Error in xtfrm.data.frame(x) : cannot xtfrm data frames

The error means `fit.OWA` is receiving a **data frame**, but inside it tries to sort / rank the rows using a method that only works on vectors or matrices, not on data frames.[163] [164]

Why this error happens

- Inside `AggWaFit718.R`, `fit.OWA` assumes its argument `the.data` is a **numeric matrix**: first n columns are inputs, last column is output.[163]
- To build an OWA, the code has to **order the inputs in each row**, which typically uses functions like `sort()` or `order()`; those rely on `xtfrm()` for ranking.[163]
- When you pass a data frame (e.g., `dataMat` is a `data.frame`), R tries to call `xtfrm.data.frame`, which is not defined → `Error in xtfrm.data.frame(x): cannot xtfrm data frames`.[164] [163]

So the fix is: **convert your data to a numeric matrix before calling** `fit.OWA`.[164] [163]

How to fix in your code

Make sure `dataMat` is a matrix, not a data frame:

```
# Read KeiHotels as matrix (recommended)
kei <- as.matrix(read.table("KeiHotels.txt", header = FALSE))

# Give names (optional but helpful)
colnames(kei) <- c(paste0("U", 1:9), "Kei")

# Use a pure numeric matrix
dataMat <- kei   # this is a matrix

# Now fit OWA
fit.OWA(
  the.data = dataMat,
  output.1 = "kei_owa_output.txt",
  stats.1  = "kei_owa_stats.txt"
)
```

If you already have `hotelData` as a data frame, convert it just before fitting:

```
dataMat <- as.matrix(hotelData)  # ensure numeric matrix
fit.OWA(
  the.data = dataMat,
  output.1 = "kei_owa_output.txt",
  stats.1  = "kei_owa_stats.txt"
)
```

Quick sanity checks

Before calling `fit.OWA`, run:

```
class(dataMat)      # should show "matrix"
is.numeric(dataMat) # should be TRUE
```

If `class(dataMat)` still shows `"data.frame"`, or if any column is not numeric, convert / clean first:

```
dataMat <- as.matrix(sapply(hotelData, as.numeric))
```

Once `the.data` is a numeric matrix, `fit.OWA` will be able to order the inputs row-wise without calling `xtfrm.data.frame`, and the error should disappear.[164] [163]

⁂

<q>read_stats ← function(file) {

s ← read.table(file, stringsAsFactors = FALSE)
out ← list(
RMSE = as.numeric(s[1, 2]),
L1 = as.numeric(s[2, 2]),
Pearson = as.numeric(s[3, 2]),
Spearman = as.numeric(s[4, 2]),
Orness = as.numeric(s[5, 2])
)
out
}

wam ← read_stats("kei_wam_stats.txt")</q> Error in scan(file = file, what = what, sep = sep, quote = quote, dec = dec, :
line 1 did not have 4 elements

3.

scan(file = file, what = what, sep = sep, quote = quote, dec = dec,
nmax = nrows, skip = 0, na.strings = na.strings, quiet = TRUE,
fill = fill, strip.white = strip.white, blank.lines.skip = blank.lines.skip,
multi.line = FALSE, comment.char = comment.char, allowEscapes = allowEscapes, ...
2.
s ← read.table(file, stringsAsFactors = FALSE)
1.
read_stats("kei_wam_stats.txt")

`read.table()` is complaining because `kei_wam_stats.txt` is **not a simple rectangular table**: different lines in the file have different numbers of "fields", so line 1 (or some later line) has fewer tokens than the maximum and `read.table()` stops with `line 1 did not have 4 elements`.[165] [166]

In AggWaFit, the stats file mixes short lines like `RMSE 7.123` with longer blocks for weights, fuzzy measures, etc., so treating the whole file as one regular table is fragile.[166] [165]

Fix 1: Safer `read.table()` (quick patch)

If you want to keep using `read.table`, add `fill = TRUE` so R pads shorter lines; this usually puts the label in column 1 and the numeric value in column 2 for the first few rows:[165] [166]

```
read_stats <- function(file) {
  s <- read.table(file, stringsAsFactors = FALSE, fill = TRUE)  # <- key change

  out <- list(
    RMSE     = as.numeric(s[1, 2]),
    L1       = as.numeric(s[2, 2]),
    Pearson  = as.numeric(s[3, 2]),
    Spearman = as.numeric(s[4,  2]),
    Orness   = as.numeric(s[5, 2])
  )
  out
}

wam <- read_stats("kei_wam_stats.txt")
wam
```

- `fill = TRUE` tells `read.table()` to pad rows with fewer fields, instead of erroring.[166] [165]
- In the AggWaFit stats files, the first few lines are consistently “label + value”, so `s[1:5, 2]` will still be the numeric values you want.[165]

If this runs without error and `wam` prints numeric values, you are good.[166] [165]

Fix 2: Robust parsing using `readLines()` (recommended)

A more robust and transparent way is to read the file line-by-line and extract the numbers you need by matching line prefixes.[165] [166]

```
read_stats <- function(file) {
  lines <- readLines(file)

  get_val <- function(pattern) {
    # Find first line starting with the pattern, e.g. "RMSE"
    idx  <- grep(pattern, lines)[^16_1]
    line <- lines[idx]
    # Extract the last number on the line
    as.numeric(sub(".*?(-?[0-9.]+)\\s*$", "\\1", line))
  }

  list(
    RMSE     = get_val("^RMSE"),
    L1       = get_val("^Av\\. abs error"),
    Pearson  = get_val("^Pearson"),
    Spearman = get_val("^Spearman"),
    Orness   = get_val("^Orness")
```

```
  )
}

wam <- read_stats("kei_wam_stats.txt")
wam
```

- `readLines()` does not care about a rectangular structure; it just gives you each line as a character string.[166] [165]
- `get_val("^RMSE")` finds the line starting with `RMSE` and pulls out the numeric value at the end, regardless of how many "columns" that line visually has.[165] [166]
- This approach will keep working even if later sections (weights, fuzzy measures) have very different formats.[166] [165]

How to debug your current file quickly

If you want to see what's actually in `kei_wam_stats.txt`, run:[165] [166]

```
readLines("kei_wam_stats.txt", n = 20)
```

You should see something like:[166] [165]

```
RMSE    7.1234
Av. abs error   5.6789
Pearson Correlation  0.95
Spearman Correlation 0.93
Orness  0.6
...
```

This confirms why a simple rectangular `read.table()` fails and why pattern-based extraction is safer.[165] [166]

Try the updated `read_stats` (Fix 1 or Fix 2), and then you can proceed to build your `error_cor_summary` table as before.[166] [165]

⁂

error_cor_summary:

From this `error_cor_summary` table, the Choquet model is clearly the best overall, the three quasi-arithmetic means are quite similar to each other, and OWA is slightly worse than the others but has a near-neutral orness.[167] [168] [169]

1. Error / correlation conclusions

Using the values you showed:

- **Choquet**: RMSE ≈ 12.83, L1 ≈ 8.18, Pearson ≈ 0.91, Spearman ≈ 0.91 → lowest errors and highest correlations, so it fits Kei's ratings best among all models.[168] [169] [167]
- **WAM, PM0.5, QM2**: RMSE ≈ 14.1–14.4, L1 ≈ 9.68–9.85, Pearson ≈ 0.887–0.890, Spearman ≈ 0.885–0.894 → very close to one another, meaning the different generators (identity, $x^{0.5}$, x^2) do not dramatically change the fit on this dataset.[167] [168]
- **OWA**: RMSE ≈ 14.73, L1 ≈ 10.30, Pearson ≈ 0.881, Spearman ≈ 0.878 → slightly higher errors and lower correlations than the quasi-arithmetic means and clearly worse than Choquet.[169] [168] [167]

Best model: Choquet, because it achieves the smallest RMSE and L1 and the largest Pearson and Spearman correlations.[168] [169]

Similar models: WAM, PM0.5, and QM2 are effectively a "cluster" with very similar error and correlation values; they form a family of quasi-arithmetic means with different generators but no huge performance difference here.[168]

Why this pattern?

- The quasi-arithmetic means (WAM, PM0.5, QM2) all assume *additive* influence of users, only differing in how they transform the ratings before averaging; if Kei's ratings are roughly additive in the users' scores, all three will end up with similar fits.[169] [168]
- OWA reorders each row and then applies a weight vector to the order statistics, so it loses the identity of individual users and focuses instead on "best/worst/middle" positions; that can hurt predictive accuracy when Kei is closer to specific users, not just generic "top-k" patterns.[167] [168]
- The Choquet integral can model *interactions* between users (synergy and redundancy) via the fuzzy measure; this extra flexibility often reduces error and increases correlation, which is exactly what you see here.[169] [167] [168]

2. Orness interpretation (from this table)

- You correctly see **NA orness** for WAM, PM0.5, QM2: orness is not naturally defined for simple quasi-arithmetic means in this implementation, so the stats files omit it.[167]
- **OWA**: Orness ≈ 0.47 → very close to 0.5, so the OWA is behaving like a fairly *balanced* average, neither strongly "max-like" (optimistic) nor strongly "min-like" (pessimistic).[168] [167]
- **Choquet**: Orness ≈ 0.49 → again near 0.5, suggesting Kei's aggregated behavior under the fitted fuzzy measure is also fairly compensatory, not dominated by either only the highest or only the lowest user ratings.[169] [167] [168]

So, from `error_cor_summary` alone you can already answer the "which model is best / which are similar / why" questions as:

- Best: Choquet.

- Similar: WAM, PM0.5, QM2.
- Different: OWA slightly worse and more neutral in orness; Choquet best and slightly more complex due to interaction modeling.[167] [168] [169]

⁂

```
# <q>dataMat <- kei</q> Please perform appropriate transformations before fitting the mod
```

The high RMSE values are mainly because you are fitting directly on the **0–100 rating scale**; an average prediction error of ~14 points is not crazy on that scale, but the textbook's workflow assumes inputs and output are first **transformed to (and sometimes smoothed)** before fitting. Using transformed data both respects the aggregation-model assumptions and makes RMSE easier to interpret.[171] [172] [173]

Below is a concrete way to transform KeiHotels and refit all models in R.

1. Load KeiHotels as before

```
kei_raw <- as.matrix(read.table("KeiHotels.txt", header = FALSE))
colnames(kei_raw) <- c(paste0("U", 1:9), "Kei")   # 9 users + Kei
```

At this point values are 0–100.[173]

2. Simple, recommended transformation: scale all columns to[171]

The aggregation functions in AggWaFit were designed for inputs in $[0,1]$ (examples in the book rescale to 0–1 before fitting), and a linear rescaling preserves order and relative distances.[172] [174]

```
# Put everything on [0,1]
kei01 <- kei_raw / 100

# Check ranges
apply(kei01, 2, range)
```

Now both the 9 user ratings and Kei's overall rating are on $[0,1]$.[173]

Set this as the matrix for fitting:

```
dataMat <- kei01
```

If you now refit, the *numerical* RMSE will drop by roughly a factor of 100 (e.g. from ≈14 to ≈0.14), because the scale changed, while correlations remain the same.[174] [172] [173]

3. Optional: monotone transformation of the output

Sometimes the output (Kei's rating) is skewed or has a nonlinear relationship to the inputs; a further monotone transform can make it more "linear" for the models. The book uses such transforms (e.g., square, square root) in some case studies.[174]

A safe choice that stays in $[0, 1]$ is a square root of the output only:

```
X <- kei01[, 1:9]          # inputs on [0,1]
y <- kei01[, 10]           # Kei on [0,1]

y_t <- sqrt(y)             # transformed output, still in [0,1]

dataMat <- cbind(X, y_t)   # this is what you pass to fit.* functions
```

Later, when you get predictions $\hat{y}_t$ from the models, you can map back to the original 0–100 scale:

```
# After fitting with dataMat, your *_output.txt files have predicted y_t at the end.
pred_t <- ...              # read last column of e.g. "kei_wam_output.txt"
pred_orig <- (pred_t^2) * 100   # back to Kei's original 0–100 scale
```

This two-step transform (0–1 scaling + sqrt) can reduce RMSE in original units if Kei's response grows nonlinearly with user ratings.[174]

If you do **not** want any nonlinearity, just stick with the simple 0–1 scaling in step 2 (no sqrt).

4. Refit all models on the transformed matrix

Once `dataMat` is defined as above (either `kei01` or `cbind(X, y_t)`), refit:

```
library(lpSolve)
source("AggWaFit718.R")

## 1) WAM
fit.QAM(
  the.data = dataMat,
  output.1 = "kei_wam_output.txt",
  stats.1  = "kei_wam_stats.txt"
)

## 2) PM0.5
fit.QAM(
  the.data = dataMat,
  output.1 = "kei_pm05_output.txt",
  stats.1  = "kei_pm05_stats.txt",
  g        = PM05,
  g.inv    = invPM05
)
```

```
## 3) QM2
fit.QAM(
  the.data = dataMat,
  output.1 = "kei_qm_output.txt",
  stats.1  = "kei_qm_stats.txt",
  g        = QM,
  g.inv    = invQM
)

## 4) OWA
fit.OWA(
  the.data = dataMat,
  output.1 = "kei_owa_output.txt",
  stats.1  = "kei_owa_stats.txt"
)

## 5) Choquet
fit.choquet(
  the.data = dataMat,
  output.1 = "kei_choquet_output.txt",
  stats.1  = "kei_choquet_stats.txt",
  k.add    = 2
)
```

Then rebuild your `error_cor_summary` using the same `read_stats` function as before; RMSE and L1 will now be in the transformed scale. If you scaled only by 100, you can simply multiply RMSE by 100 again to interpret it back in "rating points".[172] [174]

5. How to judge whether transformations "help"

After refitting:

1. **Compare RMSE and L1 in original units**
   - If you used only /100 scaling, just multiply the new RMSE by 100 and compare with your previous table.
   - If you also used sqrt on the output, convert predictions back via squaring and recompute RMSE in 0–100 units to see if it dropped.[173] [174]
2. **Check correlations**
   - Pearson / Spearman should be similar or slightly improved; big improvements suggest the transformation made the relationship more linear / monotone for the models.[174]
3. **Keep weight, orness, and fuzzy-measure interpretation**
   - These are unchanged by simple linear scaling of inputs and output; they depend on relative patterns, not absolute units. Nonlinear transforms of the output change how errors are measured but not the ordering of cases.[172] [174]

Recommended next step

- First, apply **only the 0–1 scaling** (`kei01 <- kei_raw / 100; dataMat <- kei01`), refit, and recompute `error_cor_summary`.
- If RMSE in original units (RMSE_transformed × 100) is still unsatisfactory, try the **sqrt output transform** and see whether RMSE decreases after back-transforming predictions.[173] [174]

If you paste your new `error_cor_summary` after scaling (and optionally after the sqrt transform), a concrete interpretation of how much improvement you gained and what it means for KeiHotels can be provided.[172] [173] [174]

⁂

<q>read_weights ← function(file, n_inputs = 9) {

s ← read.table(file, stringsAsFactors = FALSE)

Find the row where weights start: first row whose first column is "i"

idx ← which(s[,1] == "i")[1]
w_block ← s[(idx + 1):(idx + n_inputs),]
w ← as.numeric(w_block[, 2])
names(w) ← paste0("U", 1:n_inputs)
w
}</q> in scan(file = file, what = what, sep = sep, quote = quote, dec = dec, :
line 1 did not have 4 elements

`read.table()` is failing for the same reason as before: the stats file is **not a clean rectangular table**, so some lines have fewer "columns" than others and `scan()` throws `line 1 did not have 4 elements`.[175] [176]

In particular, AggWaFit writes the stats file as a sequence of labeled lines (RMSE, errors, correlations, then weight blocks, etc.), not as a uniform data frame, so using `read.table()` directly on the whole file is fragile.[176] [175]

Quick patch: add `fill = TRUE` (may be enough)

If you want to keep your existing approach, first try:

```
read_weights <- function(file, n_inputs = 9) {
  s <- read.table(file, stringsAsFactors = FALSE, fill = TRUE)  # <- important

  # Find the row where weights start: first row whose first column is "i"
  idx <- which(s[, 1] == "i")[^19_1]

  w_block <- s[(idx + 1):(idx + n_inputs), ]
  w <- as.numeric(w_block[, 2])
  names(w) <- paste0("U", 1:n_inputs)
```

```
  w
}
```

- `fill = TRUE` pads shorter lines with `NA` so that all rows have the same number of columns; this usually fixes the `line 1 did not have 4 elements` error.[175] [176]
- In the AggWaFit stats format, the weight block itself is simple (two entries per line: index and weight), so once `s` is read without error, `w_block[,2]` should contain the numeric weights.[175]

If this still fails or gives unexpected structure, use the more robust method below.

Robust solution: use `readLines()` and parse just the weight block

A safer way is to **read lines as plain text** and extract only the lines corresponding to the weights.[176] [175]

In the AggWaFit stats file, the weight section for WAM/PM/QM/OWA is written as:

```
i   weight i
1   0.10
2   0.15
...
9   0.12
```

[This format follows from the `cat("i","weight i")` and subsequent `cat(i, w[i])` calls in `AggWaFit718.R`.][175]

You can parse it as:

```
read_weights <- function(file, n_inputs = 9) {
  lines <- readLines(file)

  # Find the header line for the weights: starts with "i" and contains "weight"
  header_idx <- grep("^i\\s+weight", lines)[^19_1]

  # Next n_inputs lines are the weights
  w_lines <- lines[(header_idx + 1):(header_idx + n_inputs)]

  # Split each line on whitespace
  parts <- strsplit(w_lines, "\\s+")

  # First element is index, last element is weight value
  idx <- sapply(parts, function(z) as.integer(z[^19_1]))
  w   <- sapply(parts, function(z) as.numeric(z[length(z)]))

  # Name by user index (you can use paste0("U", 1:n_inputs) if you prefer)
  names(w) <- paste0("U", idx)

  w
}

# Example:
```

```
w_wam <- read_weights("kei_wam_stats.txt", n_inputs = 9)
w_wam
```

- `readLines()` avoids any assumption about a rectangular table, so it cannot fail with a "line did not have X elements" error.[176] [175]
- `grep("^i\\s+weight", lines)` finds the header line of the weight section by its text label, independent of how many columns other sections have.[175]
- Taking the next `n_inputs` lines and splitting on whitespace reliably gives you the indices and weights printed by AggWaFit.[175]

How to verify for your file

Before or after using the function, you can inspect the first part of the stats file:

```
readLines("kei_wam_stats.txt", n = 40)
```

You should see the metrics at the top (RMSE, Av. abs error, etc.), then a header like `i weight i`, then 9 lines of `index weight`.[176] [175]

Once `read_weights()` runs without error and returns a named numeric vector, you can proceed with your **Weight Analysis**: identify which users (U1–U9) have the largest weights under each model and compare them with Kei's ratings, as discussed earlier.[177] [176] [175]

⁂

read_shapley, read_fm : please correct likewise

Below are robust `read_shapley` and `read_fm` functions that, like the revised `read_weights`, use `readLines()` and simple text parsing, so they will not fail with "line did not have X elements" even though the stats files are not rectangular tables. These are tailored to the way `fit.choquet` writes its stats in `AggWaFit718.R`.[178]

1. `read_shapley`: extract Shapley indices from Choquet stats

In the Choquet stats file, the Shapley section is printed as:

```
i   Shapley i
1   0.1234
2   0.0987
...
9   0.1100
```

[written by the `cat("i","Shapley i")` and subsequent `cat(i, sh[i])` calls].[178]

This function finds that header line and reads the next `n_inputs` lines:

```
read_shapley <- function(file, n_inputs = 9) {
  lines <- readLines(file)

  # Header line: starts with "i" and contains "Shapley"
  header_idx <- grep("^i\\s+Shapley", lines)[^20_1]

  if (is.na(header_idx)) {
    stop("Could not find Shapley header (line starting with 'i Shapley') in file: ", file
  }

  # Next n_inputs lines contain: index and Shapley value
  shap_lines <- lines[(header_idx + 1):(header_idx + n_inputs)]

  parts <- strsplit(shap_lines, "\\s+")
  idx   <- sapply(parts, function(z) as.integer(z[^20_1]))
  vals  <- sapply(parts, function(z) as.numeric(z[length(z)]))

  names(vals) <- paste0("U", idx)
  vals
}

# Example:
# sh_chq <- read_shapley("kei_choquet_stats.txt", n_inputs = 9)
# sh_chq
```

- Because this works line-by-line, it is insensitive to how many "columns" later blocks (like fuzzy measures) have.[178]
- The names `U1, ..., U9` correspond to your nine users; adjust naming if your variables are named differently.[179]

2. `read_fm`: extract fuzzy measure m(A) values

The fuzzy measure block is printed as:

```
binary number   fm.weights
1   0.05
2   0.07
...
511 1.00
```

Here the first column is the integer index of the subset A, interpreted in binary (bit i = user i).[178]

This function reads that entire block and returns a numeric vector `fm` of length 2^n, with `fm[k+1] = m(A_k)` and `fm[^20_1]` unused (since index 0 is empty set):

```
read_fm <- function(file, n_inputs = 9) {
  lines <- readLines(file)

  # Header line for fuzzy measure: starts with "binary number"
```

```
  header_idx <- grep("^binary\\s+number", lines)[^20_1]

  if (is.na(header_idx)) {
    stop("Could not find fuzzy measure header (line starting with 'binary number') in fi
  }

  # All lines after header are of the form: index value
  fm_lines <- lines[(header_idx + 1):length(lines)]

  # Drop any empty lines
  fm_lines <- fm_lines[nzchar(fm_lines)]

  parts <- strsplit(fm_lines, "\\s+")
  idx   <- sapply(parts, function(z) as.integer(z[^20_1]))
  vals  <- sapply(parts, function(z) as.numeric(z[length(z)]))

  # Initialize fm[0..2^n - 1] as numeric; we store in positions k+1
  fm <- numeric(2^n_inputs)
  fm[idx + 1] <- vals

  fm
}

# Example:
# fm <- read_fm("kei_choquet_stats.txt", n_inputs = 9)
# length(fm)      # 512 for 9 inputs
# fm[2^(1:9) + 1] # singleton measures m({Ui})
```

- `idx` runs from 1 to $2^n - 1$; the full set (all users) has index $2^n - 1$ and measure 1.[178]
- You can then define a helper `m_set()` exactly as before to query specific subsets:[178]

```
m_set <- function(vars, fm) {
  # vars: vector of user indices, e.g. c(1,3) for U1 and U3
  idx <- sum(2^(vars - 1))
  fm[idx + 1]
}
```

3. Using these functions for your analyses

Once these are in place, you can do:

```
sh_chq <- read_shapley("kei_choquet_stats.txt", n_inputs = 9)
fm_chq <- read_fm("kei_choquet_stats.txt", n_inputs = 9)

sh_chq   # Shapley "weights" per user
m_set(c(1), fm_chq)    # m({U1})
m_set(c(1,2), fm_chq)  # m({U1,U2})
```

Then your pairwise additivity/superadditivity/redundancy code from earlier works unchanged:

```
n_inputs <- 9
pair_analysis <- data.frame()
for (i in 1:(n_inputs - 1)) {
  for (j in (i + 1):n_inputs) {
    m_i  <- m_set(i, fm_chq)
    m_j  <- m_set(j, fm_chq)
    m_ij <- m_set(c(i, j), fm_chq)

    rel <- if (abs(m_ij - (m_i + m_j)) < 1e-6) {
      "additive"
    } else if (m_ij > m_i + m_j) {
      "superadditive"
    } else {
      "subadditive (redundant)"
    }

    pair_analysis <- rbind(
      pair_analysis,
      data.frame(Ui = i, Uj = j, m_i = m_i, m_j = m_j, m_ij = m_ij, relation = rel)
    )
  }
}
pair_analysis
```

With `read_shapley` and `read_fm` corrected this way, you can now safely complete your Shapley-based weight analysis, orness analysis (from the stats header), and fuzzy-measure interaction analysis for the Choquet model without running into `scan`/`read.table` errors.[180] [179] [178]

⁂

```
# <q>For Choquet, the orness reflects the underlying fuzzy measure and tells you whether
```

For a Choquet model, the orness number is a global summary of the **fuzzy measure** v that tells whether the model behaves more like taking a maximum over "high" user ratings (optimistic / OR-like) or like requiring many users to agree (pessimistic / AND-like).[181] [182] [183] [184]

Choquet integral and fuzzy measure

- The Choquet integral $C_v(x)$ aggregates a vector of inputs $x = (x_1, \ldots, x_n)$ using a fuzzy measure v defined on **all subsets** of users, not just on single users.[185] [183] [181]
- The fuzzy measure $v(A)$ captures how important the whole *coalition* A is; for example $v(\{U_3, U_7\})$ can be larger, equal to, or smaller than $v(\{U_3\}) + v(\{U_7\})$, representing synergy, additivity, or redundancy between those two users.[183] [181] [185]

How orness is defined for a fuzzy measure

- For simple averaging operators like OWAs, orness is defined from the weight vector and measures how close the operator is to a max versus a min.[182] [184]
- For a Choquet integral, AggWaFit uses a function `orness.v(v)` that takes the fuzzy measure values and computes a weighted average of $v(A)$ over subsets of different sizes, with larger subsets getting higher multipliers.[184] [183]
- Conceptually, orness is close to

$$\text{orness}(v) \approx \frac{1}{n-1} \sum_{k=1}^{n} (k-1)\, \overline{v_k},$$

  where $\overline{v_k}$ is the average value of $v(A)$ over all coalitions A of size k; this increases when larger coalitions tend to have much higher measure than small ones.[185] [182] [183]

"Coalitions of high raters" in this context

- When you plug user ratings $x_1, \ldots, x_9$ into the fitted Choquet model, the integral decomposes the contribution of each **subset of users** through terms like $(x_{(i)} - x_{(i-1)})\, v(A_i)$, where A_i contains the users with at least as large a rating as the i-th ordered user.[181] [183] [185]
- If the fuzzy measure v assigns large values to coalitions that contain at least one very high rater (for example sets that include a "top" user), then any time that user rates a hotel highly the Choquet integral will increase strongly; this is an *optimistic*, OR-like behavior with higher orness.[182] [183] [181]
- If instead v stays small unless **several users together** form a coalition, then the Choquet integral only becomes large when many users rate a hotel highly; this is a more *conservative*, AND-like behavior with lower orness.[183] [185] [182]

Interpreting orness values for Choquet

- Orness near 1: larger coalitions dominate the fuzzy measure, meaning that "it is enough that some strong subset is high" for Kei's predicted rating to be high, which is similar to an OR over users.[184] [185] [182] [183]
- Orness near 0: small coalitions (or even the weakest users) dominate, so the Choquet integral behaves more like an AND or a minimum—any low rating can hold Kei's predicted rating down.[185] [182] [183] [184]
- Orness near 0.5: the measure is balanced across subset sizes, so the Choquet integral behaves more like a compensatory average; good ratings can offset bad ones, but neither single extremes nor single weak ratings dominate.[182] [183] [184] [185]

What this means for your KeiHotels Choquet model

- In your `error_cor_summary` the Choquet orness is around 0.49, which is very close to 0.5; that indicates Kei's fitted Choquet model neither behaves like a pure max nor like a pure min across the users.[186] [183] [184]

- Intuitively, this tells you that Kei's ratings are modelled as **moderately compensatory**: high ratings from some users can partly compensate for lower ratings from others, but there is no strong "just one high user is enough" effect, and no strong "everyone must agree" effect. [186] [181] [183] [184] [182]
- When you look at the fuzzy measure values and Shapley indices, pairs or groups of users for which $v(A)$ is much larger than the sum of singletons contribute to that behaviour; orness takes the entire pattern of these coalition weights and summarizes whether those strong coalitions tend to involve many users (more OR-like) or require broad agreement (more AND-like). [181] [183] [184] [186] [185]

So, "for Choquet, the orness reflects the underlying fuzzy measure and tells you whether coalitions of high raters are treated more optimistically or conservatively" means: look at how the fuzzy measure distributes importance across *sizes* of user coalitions; the orness number is the one-dimensional summary of that distribution, indicating whether the integral rewards the presence of any strong coalition (optimistic) or demands broad agreement among users (conservative) to produce a high predicted rating. [183] [184] [186] [181] [185] [182]

⁂

how to import 'R' notebook into workspace in posit cloud

In Posit Cloud (RStudio in the browser), an "R notebook" is just an `.Rmd` file with notebook options turned on. To use an existing notebook in a project, you simply upload the `.Rmd` (or a zip/Git repo containing it) into that project. [190] [191]

Upload a local R Notebook file (.Rmd)

1. Open your Posit Cloud project. [192] [193]
2. In the **Files** pane (bottom right), click **Upload**. [194] [190]
3. In the dialog, choose your R notebook file, e.g. `analysis.Rmd` (or `something.nb.html` plus its `.Rmd` source if you have both). [195] [190]
4. Click **OK/Upload**; the file will appear in the Files pane. [196] [190]
5. Click the file name (e.g., `analysis.Rmd`) to open it in the RStudio editor; you can now run chunks, knit, etc. [197] [198]

If you have several related files (data, scripts, notebook), you can zip them first and upload the zip; Posit Cloud will automatically unpack it in the current folder. [191] [190]

Import from a GitHub repository (if the notebook is on Git)

1. On the Posit Cloud home page, click the **New Project** dropdown. [193] [191]
2. Choose **New Project from Git Repository**. [191]
3. Paste the repository URL containing your R notebook(s). [191]

4. Posit Cloud will create a new project with all tracked files (including `.Rmd` notebooks) already in the workspace.[190] [191]

Then open the `.Rmd` file in the Files pane as usual.[198] [197]

Convert other notebook formats to R Markdown

If your "R notebook" is from another environment (e.g., Databricks, Jupyter), export it to R Markdown first, then upload the `.Rmd` into Posit Cloud as above.[199] [195]

In summary: create/open a project, use the **Files** → **Upload** button (or "New Project from Git Repository") to bring the `.Rmd` into that project, then click the file to work with it as an R notebook.[193] [190] [191]

⁂

How to upload an exported Rmd file into a Posit Cloud project

To bring an exported `.Rmd` file into a Posit Cloud project, you just upload it through the Files pane of that project.[210] [211] [212]

Step-by-step in an existing project

1. Open your Posit Cloud project (click its name on the Posit Cloud home page).[213] [214]
2. In the lower-right pane, click the **Files** tab (it shares a pane with Plots, Packages, Help, Viewer).[211] [210]
3. Click the **Upload** button at the top of the Files pane.[215] [216] [210]
4. In the dialog that opens, click **Choose File** (or similar) and select your exported `.Rmd` file from your computer.[210] [211]
5. Click **OK/Upload**. The `.Rmd` will appear in the Files list in your project's working directory.[212] [211]
6. Click on the `.Rmd` file name in the Files pane to open it in the RStudio source editor; you can now edit it, run chunks, or click **Knit**.[217] [218] [219]

Uploading multiple files or a whole project

- If your notebook depends on other files (data, images, scripts), zip them together, then:[211] [212] [210]
  - Use **Files** → **Upload**, select the `.zip`, and upload.
  - Posit Cloud automatically unzips the archive into the current directory.
- Alternatively, if the `.Rmd` is in a GitHub repo, you can create a **New Project from Git Repository** using the repo URL; the notebook will appear automatically in that project.[220] [212]

So the core action is: open your project → go to the **Files** tab → click **Upload** → choose your `.Rmd` → open it in the editor.[212] [210] [211]

⁂

'R' library : library(lpSolve)

method: lp
role : 'R' data engineer
context: I need to transform variable x such that it's distribution would be turned from skewed to normal. For this purpose I am applying x^n transformation. Here I need to identify optimum value of n.

task : to device a method in 'R' which would take x as parameter and use lp method to find optimum value of n.

The usual way to choose a power n that makes x^n "as normal as possible" is to treat this as a **nonlinear, 1-D search problem**, not as a linear program; `lpSolve::lp()` is designed for linear objectives in linear constraints and is not a natural fit here.[230] [231]

Below is a practical, engineer-style solution in R that mimics what you want (optimize over n), but using **grid search or** `optimize()` plus a normality criterion such as Shapiro–Wilk p-value or absolute skewness.

1. Define a normality score for a given n

Example 1: maximize Shapiro–Wilk p-value of x^n (higher p → closer to normal).[232] [233] [234]

```
normality_score_sw <- function(x, n) {
  # Transform; guard against negatives / zeros
  x_t <- x ^ n
  # Shapiro–Wilk; return p-value
  shapiro.test(x_t)$p.value
}
```

Example 2: minimize absolute skewness of x^n.[234]

```
library(e1071)  # for skewness()

normality_score_skew <- function(x, n) {
  x_t <- x ^ n
  abs(skewness(x_t, na.rm = TRUE))
}
```

2. Grid search over n

For power transforms, a reasonable range is, e.g., $n \in [-2, 2]$ or whatever makes sense for your data domain (ensuring x^n is defined).[235] [234]

Maximize Shapiro p-value:

```
find_n_shapiro <- function(x, n_grid = seq(-2, 2, by = 0.05)) {
  scores <- sapply(n_grid, function(n) normality_score_sw(x, n))
  best_i <- which.max(scores)
  list(
    n_opt   = n_grid[best_i],
    p_opt   = scores[best_i],
    grid    = n_grid,
    scores  = scores
  )
}
```

Minimize absolute skewness:

```
find_n_skew <- function(x, n_grid = seq(-2, 2, by = 0.05)) {
  scores <- sapply(n_grid, function(n) normality_score_skew(x, n))
  best_i <- which.min(scores)
  list(
    n_opt   = n_grid[best_i],
    skew_opt = scores[best_i],
    grid    = n_grid,
    scores  = scores
  )
}
```

Usage:

```
res_sw   <- find_n_shapiro(x)
res_skew <- find_n_skew(x)

res_sw$n_opt
res_skew$n_opt

x_opt <- x ^ res_sw$n_opt    # transformed variable
```

3. Continuous optimization with `optimize()`

If you want a smoother 1-D search:

```
best_n_skew <- function(x, lower = -2, upper = 2) {
  f <- function(n) normality_score_skew(x, n)
  optimize(f, interval = c(lower, upper))  # minimizes f
}

opt_res <- best_n_skew(x)
```

```
opt_res$minimum       # n_opt
opt_res$objective     # min abs skewness
```

4. Why `lpSolve::lp()` is not appropriate here

- `lp()` solves problems of the form $\min c^\top z$ subject to $Az < b$, with **linear** objective and constraints in the decision variables z. [236] [230]
- Your decision variable is a single scalar n, and the "objective" is something like *Shapiro p-value of* x^n or *abs(skewness of* x^n*)*—these are **highly nonlinear** in n. [232] [234]
- There is no natural linear constraint system to express "make x^n normal" in terms of n; any realistic formulation would require nonlinear optimization, not linear programming. [231] [236]

So even though you are using `lpSolve` elsewhere for weight fitting (AggWaFit), for **normality-driven power transforms** the right tools are **Box–Cox-style methods**, `optimize()`, or grid search over n, not `lp()`. [237] [238] [239] [235]

5. If you want an "official" power transform

R already has well-tested functions for this:

- `MASS::boxcox(lm(x ~ 1))` to visually choose λ. [238] [240]
- `forecast::BoxCox.lambda(x, method = "loglik")` to estimate the best λ numerically. [239] [241]
- `AID::boxcoxnc()` / `bestNormalize::boxcox()` for more advanced options. [242] [241]

These effectively find the optimal exponent (Box–Cox λ) to make data as normal as possible, which is the same spirit as your x^n search, but using maximum likelihood instead of linear programming. [243] [237] [242] [235]

If you share a sample vector `x`, a ready-to-run function that tries both Shapiro and skewness criteria and returns the suggested n plus diagnostic plots can be provided.

⁂

"shapiro.test(x_t)$p.value" **what is the 'R' package ?**

`shapiro.test()` is part of base R, in the **stats** package. [250] [251]

- You do **not** need to install anything; the `stats` package is loaded by default when R starts, so `shapiro.test(x_t)$p.value` works out of the box. [252] [250]
- You can confirm by checking `?shapiro.test` in the console; the help page header will show `package: stats`. [250] [252]

⁂

<q>find_n_shapiro ← function(x, n_grid = seq(-2, 2, by = 0.05)) {

scores ← sapply(n_grid, function(n) normality_score_sw(x, n))
best_i ← which.max(scores)
list(
n_opt = n_grid[best_i],
p_opt = scores[best_i],
grid = n_grid,
scores = scores
)
}</q> Error in shapiro.test(x_t) : all 'x' values are identical

The error means that for some power n in your grid, the transformed data x^n became **constant**, and the Shapiro–Wilk test cannot run on a vector where all values are identical.[259] [260]

Why this happens

- If your original x has only a few distinct values (or includes zeros/ones), then for extreme powers (e.g., very large positive or negative n), many or all values of x^n can collapse to the same number.[261] [262]
- Shapiro–Wilk requires some variability; if `length(unique(x_t)) == 1`, it throws `all 'x' values are identical`.[260] [259]

Fix: skip invalid n (or return NA)

Wrap the call to `shapiro.test()` in a safe helper that returns `NA` if the vector is constant (or has too few unique values), then ignore those `NA` scores when searching for the best n.[259] [260] [261]

```
normality_score_sw <- function(x, n) {
  x_t <- x ^ n
  # Remove NAs just in case
  x_t <- x_t[is.finite(x_t)]
  # Need at least 3 distinct values for a sensible test
  if (length(unique(x_t)) < 3) return(NA_real_)
  shapiro.test(x_t)$p.value
}

find_n_shapiro <- function(x, n_grid = seq(-2, 2, by = 0.05)) {
  scores <- sapply(n_grid, function(n) normality_score_sw(x, n))
  # Exclude NA scores
  valid  <- !is.na(scores)
  if (!any(valid)) stop("No valid n in the grid (all transforms constant or invalid).")
  best_i <- which.max(scores[valid])
  list(
    n_opt   = n_grid[valid][best_i],
    p_opt   = scores[valid][best_i],
    grid    = n_grid,
    scores  = scores
```

```
  )
}
```

Usage

```
res_sw <- find_n_shapiro(x)
res_sw$n_opt   # optimal n
res_sw$p_opt   # corresponding Shapiro p-value
```

If your data have many repeated values or a very narrow range, you can also restrict the grid (e.g., `seq(-1, 1, by = 0.05)`) to avoid extreme powers that tend to produce constant vectors. [262] [261]

⁂

Attached are the performance metrics of various aggregation models; Please analyze and suggest the model with optimal metrics

From the three stats files you shared, the power mean and WAM behave similarly and the OWA achieves the lowest errors but with very weak correlations and an extreme weight pattern; overall, the **power mean** looks like the best compromise given your "low error + correlations close to 1" criterion. [263] [264] [265]

1. Metrics comparison

Model	RMSE	L1 (avg abs error)	Pearson	Spearman	Orness
WAM	0.4619	0.4373	0.1096	0.2986	NA
Power mean	0.4564	0.4340	0.1103	0.2888	NA
OWA	0.3237	0.2975	0.0467	0.1101	0.0000

[27_2][27_3]

- **Errors:** OWA has the smallest RMSE (0.324) and L1 (0.298), clearly better than WAM (RMSE 0.462, L1 0.437) and power mean (RMSE 0.456, L1 0.434). [265] [263] [264]
- **Correlations:** WAM and power mean have similar and modest Pearson/Spearman correlations (~0.11–0.30), while OWA correlations are much weaker (Pearson ≈ 0.047, Spearman ≈ 0.11). [263] [264] [265]

2. Weight behavior

- **WAM:** weights ≈ (0.876, 0.124, 0, 0, 0): almost all weight on variable 1, some on variable 2, others ignored. [264]
- **Power mean:** weights ≈ (0.805, 0.195, 0, 0, 0): similar story—variables 1 and 2 dominate. [263]

- **OWA:** weights = (1, 0, 0, 0, 0) with orness = 0, i.e., the fitted OWA has collapsed to a single input and behaves like a pure min/"AND" on that ordered position (in practice, one statistic per row), ignoring all other information.[265]

This explains why OWA can get low absolute error (it is effectively using a single extremal statistic that happens to approximate the target numerically) but has poor correlation and no real multi-criteria structure.[264] [265] [263]

3. Recommendation

Given your criteria ("low error values and correlations close to 1" from the textbook instructions):

- **Discard / treat OWA with caution**: despite the smallest errors, its correlations are very low and the weight vector is degenerate (all weight on one component), which usually indicates over-simplification or overfitting to a particular statistic rather than a robust aggregation of all variables.[265] [263] [264]
- **WAM vs power mean:** errors and correlations are very close; the **power mean** has slightly better RMSE and L1, while WAM has slightly better Spearman correlation, but differences are tiny.[263] [264]

So, among these three, the **power mean model** is the best overall choice: it gives the lowest errors among the "well-behaved" models and correlations comparable to WAM, while using a similar, interpretable weight pattern across variables.[264] [265] [263]

⁂

you have skipped choquet integral i.e. fuzzy weights analysis

You are right, the Choquet integral deserves its own discussion. With your `choquet-stats3.txt`, the Choquet model is effectively behaving almost identically to the OWA you fitted: same errors, same correlations, orness 0, and a very extreme fuzzy measure.[266] [267] [268] [269]

1. Choquet performance vs other models

From `choquet-stats3.txt`:

- RMSE = 0.323726470160389
- Av. abs error = 0.297507726705669
- Pearson Correlation = 0.0467440565184006
- Spearman Correlation = 0.11006829419423
- Orness = 0

[266]

These are **identical** to the OWA metrics in `OWA-stats3.txt`:[269] [266]

- RMSE = 0.323726470160389

- Av. abs error = 0.297507726705669
- Pearson Correlation = 0.0467440565184006
- Spearman Correlation = 0.11006829419423
- Orness = 0

So, with the current data and constraints, the fitted Choquet integral has essentially collapsed to the same shape as your OWA: very low error but very weak correlations and purely "AND-like" behavior (orness 0).[267] [268] [269] [266]

2. Shapley values (fuzzy "weights")

The Shapley block is:[266]

- i Shapley i
  - 1 → 0.2
  - 2 → 0.2
  - 3 → 0.2
  - 4 → 0.2
  - 5 → 0.2

This means:

- All five inputs have **equal Shapley indices** (0.2 each), so in terms of *overall marginal contribution across all coalitions*, the Choquet model says every variable is equally important.[266]
- This contrasts with WAM / power mean weights, which strongly favor variable 1 (and to a lesser extent 2) and give zero to others.[268] [267]

So from a "weight analysis" perspective:

- **Choquet** is the most *balanced* model: no variable is privileged, all have equal Shapley importance.[266]
- **WAM** and **power mean** are highly skewed: almost all weight on variable 1 (and a bit on 2), others effectively ignored.[267] [268]

3. Fuzzy measure pattern (additivity / interaction)

The fuzzy measure section:[266]

- binary number fm.weights
  - 1...30 → 0
  - 31 → 1

For 5 inputs, index 31 (binary 11111) is the **full set of all 5 variables**, and all other subsets (singletons, pairs, triples, quadruples) have measure 0.[266]

Implications:

- For any **proper subset** A of variables, $m(A) = 0$. Only the full coalition $A = \{1, 2, 3, 4, 5\}$ has $m(A) = 1$.[266]
- This is an extreme **superadditive / AND-like** measure:
  - $m(\{i\}) = 0$ for every i,
  - $m(\{i, j\}) = 0 < m(\{i\}) + m(\{j\}) = 0$ (degenerate),
  - but $m(\{1, 2, 3, 4, 5\}) = 1 \gg \sum_i m(\{i\}) = 0$.[266]
- In terms of synergy/redundancy:
  - All proper subsets have no effect individually; only when *all 5* variables are "high together" does the Choquet integral output become large.[266]

This explains:

- **Orness = 0**: the measure puts all mass on the "largest coalition size" and none on smaller ones, producing the most AND-like possible Choquet integral.[266]
- Very low correlations: because the model only "fires" when all ratings are simultaneously strong, it cannot track more nuanced variation across cases, even though it can keep the average absolute error numerically small.[269] [267] [266]

4. How to interpret Choquet vs other models here

Putting it together:

- **Choquet's error metrics** match OWA (lowest RMSE/L1 among your models) but its correlations are very weak, so it does *not* capture the ranking / pattern of the true output well, just its approximate level.[267] [269] [266]
- **Shapley values** say "all variables matter equally," but the **fuzzy measure** says "only the full coalition actually has any standalone effect," which is an extremely strong logical AND.[266]
- Compared to WAM / power mean:
  - WAM / power mean: modest errors, better correlations, but highly skewed weights towards one or two variables.[268] [267]
  - Choquet: very balanced Shapley weights but a degenerate, coalition-only measure and poor correlation.[268] [267] [266]

5. Recommendation including Choquet

Considering **all four** models (WAM, power mean, OWA, Choquet):

- If the priority is **predictive fit plus understandable structure** (low error *and* decent correlations), the **power mean (or WAM)** still look better: they have reasonable errors and significantly higher Pearson/Spearman correlations, despite skewed weights.[267] [268]
- The **Choquet integral**, as fitted here, is *too extreme*:
  - Orness = 0, only full coalition has measure 1, correlations extremely low.[266]
  - It suggests a pure "all must agree" rule, which is usually unrealistic for rating aggregation.[266]

So the fuzzy-weights analysis shows that your Choquet fit is mathematically valid but **not a good descriptive model** of the data's structure. In this scenario, it is better to rely on the **power mean** (or WAM) as your primary model and treat the Choquet result as an illustration of an extreme AND-like aggregation rather than the optimal choice.[268] [267] [266]

⁂

consider this also

Including `powermean-stats31.txt` does not change the earlier conclusion: the power mean remains the best compromise among the nondegenerate models, and the Choquet/OWA pair remain extreme, AND-like models with poor correlations despite low errors.[270] [271] [272] [273] [274]

Updated view with the extra power-mean file

The new `powermean-stats31.txt` has:[270]

- RMSE = 0.45635902165036
- Av. abs error = 0.433986670793403
- Pearson correlation = 0.110251793295464
- Spearman correlation = 0.288792852920447
- Weights: (0.8050, 0.1950, 0, 0, 0)

These are **identical** to the earlier `powermean-stats3.txt` metrics and weights:[271] [270]

- RMSE ≈ 0.45636
- L1 ≈ 0.43399
- Pearson ≈ 0.11025
- Spearman ≈ 0.28879
- $w = (0.8050, 0.1950, 0, 0, 0)$

So this second file is just a repeat/alternative name for the same power-mean fit, confirming that model's behavior.

Final comparative picture

Summarizing all four models:[272] [273] [274] [271] [270]

- **WAM**
  - RMSE ≈ 0.4619, L1 ≈ 0.4373
  - Pearson ≈ 0.1096, Spearman ≈ 0.2986
  - Weights ≈ (0.876, 0.124, 0, 0, 0)
- **Power mean (p-model)**
  - RMSE ≈ 0.4564, L1 ≈ 0.4340

- Pearson ≈ 0.1103, Spearman ≈ 0.2888
- Weights ≈ (0.805, 0.195, 0, 0, 0)

- **OWA**
  - RMSE ≈ 0.3237, L1 ≈ 0.2975
  - Pearson ≈ 0.0467, Spearman ≈ 0.1101
  - Orness = 0, OWA weights = (1, 0, 0, 0, 0)
- **Choquet**
  - RMSE ≈ 0.3237, L1 ≈ 0.2975
  - Pearson ≈ 0.0467, Spearman ≈ 0.1101
  - Orness = 0
  - Shapley: all 0.2 (balanced marginal importance)
  - Fuzzy measure: all subsets have measure 0 except the full set (index 31) with measure 1 → extreme AND-like coalition.

Recommendation (with everything considered)

- **Best overall balance:** Power mean
  - Slightly lower RMSE/L1 than WAM, similar correlations, and a clear, interpretable weight pattern (variables 1 and 2 important, others negligible).[271] [272] [270]
- **WAM:** Very close second, almost identical performance to power mean; choosing between them is largely a stylistic or theoretical preference.[272]
- **OWA and Choquet:**
  - Mathematically lowest errors, but correlations are very weak and both models have degenerate, highly AND-like behavior (orness 0; Choquet's fuzzy measure only values the full coalition, OWA uses only one position).[273] [274]
  - Good examples of extreme aggregation, but not good descriptive models of the data's pattern.

So after adding `powermean-stats31.txt`, the **power mean** remains the preferred model under your criteria (low error, reasonable correlations, and interpretable weights), with WAM similar and OWA/Choquet clearly too extreme despite their low numerical error.[274] [273] [270] [271] [272]

⁂

plz consider this also

Including `powermean-stats23.txt` confirms the earlier story: among all the models and power-mean variants you shared, the **power mean with stats matching** `powermean-stats3` **/** `powermean-stats31` is still the best compromise; `powermean-stats23` is weaker on all metrics.[275] [276] [277] [278] [279] [280]

What the new file shows

`powermean-stats23.txt`:[275]

- RMSE = 0.4661
- Av. abs error = 0.4454
- Pearson = 0.0962
- Spearman = 0.2592
- Weights ≈ (0.718, 0.282, 0, 0, 0)

Compare to the "better" power mean (`powermean-stats3` / `31`):[276] [280]

- RMSE = 0.4564 (smaller → better)
- Av. abs error = 0.4340 (smaller)
- Pearson = 0.1103 (larger)
- Spearman = 0.2888 (larger)
- Weights ≈ (0.805, 0.195, 0, 0, 0)

So `powermean-stats23` has:

- **Higher** RMSE and L1 (worse fit).[280] [276] [275]
- **Lower** correlations (worse association with true output).[276] [280] [275]

It just represents another power-mean fit with a slightly different trade-off between user 1 and user 2 (more weight on user 2), but overall performance is **inferior** to your earlier power-mean model.[280] [275] [276]

Overall ranking (all models and this extra file)

Putting everything together:[277] [278] [279] [275] [276] [280]

- **Best nondegenerate model:**
  - Power mean (`powermean-stats3` / `31`): lowest error among the "reasonable" models, reasonable (though not huge) correlations, interpretable weights concentrated on variables 1 and 2.
- **Very similar alternative:**
  - WAM: slightly worse error but similar correlations and similar weight pattern; acceptable if you prefer the arithmetic mean form.[277]
- **Weaker power-mean variant:**
  - `powermean-stats23`: strictly worse on both errors and correlations than the above power-mean, so not preferred.[275]
- **Extreme models:**
  - OWA and Choquet: lowest RMSE/L1 but very low correlations and highly AND-like behavior (orness 0; OWA weight; Choquet measure that only values the full coalition).

These are informative as extreme cases but not good descriptive or predictive models. [278] [279] [281]

So even after considering `powermean-stats23.txt`, the recommendation remains: **use the power mean with stats in `powermean-stats3` / `powermean-stats31` as your main model**, keep WAM as a close benchmark, and treat the other power-mean variant, OWA, and Choquet primarily as comparison/teaching examples rather than optimal choices. [279] [278] [276] [277] [280] [275]

⁂

This is new set of model metrics. Please evaluate for optimal model among these

Across this new set, the **power-mean model in `powermean-stats31.txt` (same metrics as `powermean-stats3.txt`) is still the optimal choice**, with WAM very close behind and OWA/Choquet clearly too extreme despite low error. [282] [283] [284] [285] [286] [287]

Metrics summary for all models

From the latest stats files:

- **WAM (`WAM-stats3.txt`)** [283]
  - RMSE ≈ 0.4619
  - Av. abs error ≈ 0.4373
  - Pearson ≈ 0.1096
  - Spearman ≈ 0.2986
  - Weights ≈ (0.876, 0.124, 0, 0, 0)
- **Power mean (model A: `powermean-stats3.txt` / `powermean-stats31.txt`)** [286] [282]
  - RMSE ≈ 0.4564
  - Av. abs error ≈ 0.4340
  - Pearson ≈ 0.1103
  - Spearman ≈ 0.2888
  - Weights ≈ (0.805, 0.195, 0, 0, 0)
- **Power mean (model B: `powermean-stats23.txt`)** [287]
  - RMSE ≈ 0.4661
  - Av. abs error ≈ 0.4454
  - Pearson ≈ 0.0962
  - Spearman ≈ 0.2592
  - Weights ≈ (0.718, 0.282, 0, 0, 0)
- **OWA (`OWA-stats3.txt`)** [284]
  - RMSE ≈ 0.3237

- Av. abs error ≈ 0.2975
- Pearson ≈ 0.0467
- Spearman ≈ 0.1101
- Orness = 0
- OWA weights = (1, 0, 0, 0, 0)

- **Choquet (`choquet-stats3.txt`)**[285]
  - RMSE ≈ 0.3237
  - Av. abs error ≈ 0.2975
  - Pearson ≈ 0.0467
  - Spearman ≈ 0.1101
  - Orness = 0
  - Shapley: all 0.2 (balanced marginal importance)
  - Fuzzy measure: all subsets 0 except full set (index 31) = 1

Which model is optimal?

Given your stated criteria (low error *and* correlations as close to 1 as possible):

1. **Discard the clearly worse power mean (model B)**
   - `powermean-stats23` has higher RMSE/L1 and lower correlations than model A and WAM. [282] [283] [287]
2. **Treat OWA and Choquet as extreme cases**
   - They have the lowest RMSE/L1 but *very* low correlations and degenerate behavior:
     - OWA: all weight on a single order position, orness 0.[284]
     - Choquet: only the full coalition has measure 1, orness 0, purely AND-like.[285]
   - These are useful for illustrating extreme aggregation but not for balanced modelling of your data.
3. **Compare WAM vs power-mean model A**
   - Errors: power mean A has slightly lower RMSE and L1 than WAM.[283] [282]
   - Correlations: WAM has marginally higher Spearman; Pearson is almost identical.[282] [283]
   - Both strongly favor variables 1 and 2, but the power mean distributes a bit more weight to user 2 (0.195 vs 0.124), which may better reflect the data.[286] [283] [282]

Considering all of this, **power-mean model A (`powermean-stats3` / `powermean-stats31`) is the optimal model among this set**, with WAM as a very close benchmark; OWA and Choquet are extreme AND-like aggregators with poor correlation and should not be chosen as the main predictive/analytic model here.[287] [283] [284] [285] [286] [282]

⁂

how to spawn new conversion from this existing conversation? it is continuation of this aggregation model conversion

To “spawn” a new conversation that still builds on this aggregation-models work, do either of the following:

1. In this tool (chat interface)

- Start typing your **new question directly here**, but **summarize the relevant context in the message itself**, for example:
  - “New sub-task (same KeiHotels aggregation project): now I want to compare the optimal power-mean model vs a regression baseline. Here are the stats we had: …”
- The assistant will then treat it as a fresh sub-thread but with the context you just restated.
- This is usually enough; you do **not** need a special button—just make the first line say something like “New thread for: …” so you can recognize it later.

2. In Posit Cloud / R notebook (for code)

If you mean a new *code workflow* rather than chat:

- Create a **new R script or R Markdown** (File → New File → R Script / R Markdown).
- At the top, briefly re-document the context (dataset, best model, key stats) and then continue with new analysis code (e.g., new models, plots, diagnostics).

If you tell what the next step is (for example, “compare power mean vs linear regression” or “visualize residuals for the best model”), a clean “first cell” / first message for that new thread can be drafted for you.

<q>1. In this tool (chat interface)

Start typing your new question directly here, but summarize the relevant context in the message itself, for example:

“New sub-task (same KeiHotels aggregation project): now I want to compare the optimal power-mean model vs a regression baseline. Here are the stats we had: …”

The assistant will then treat it as a fresh sub-thread but with the context you just restated.

This is usually enough; you do not need a special button—just make the first line say something like “New thread for: …” so you can recognize it later.</q> that's fine, however it's been slow to open this conversation

Long chats can get slow because every new reply has to load and process the entire history, so what you’re seeing is expected once a thread gets large.[288] [289] [290]

To keep working on the same aggregation project **without** the lag:

1. Start a fresh chat, carry only a short summary

- In this interface, click **New chat** (or equivalent) to open a blank conversation.[290]
- Paste a **compact summary** as your first message, for example:
  - "New thread for KeiHotels aggregation project: we fitted WAM, power mean (best so far), OWA, and Choquet with these metrics and weights: … Now I want to do X."
- Then continue with new questions there; performance will be much faster because the context is short.[289] [288] [290]

2. Reuse your own written notes

- You can also keep a small text/Markdown file in Posit Cloud or locally that summarizes: dataset, transformations, best models, key stats.
- When you start a new chat, copy-paste only the parts needed for the next sub-task (e.g., "best model is power mean with RMSE ~0.456, weights (0.80,0.20,0,0,0)…").[288] [289]

If you like, the next message you send can be "New thread for KeiHotels: [short recap] + new question", and a concise recap text for you to paste into that new chat can be drafted.

⁂

WHAT IS The condition number in VIF analysis

The **condition number** in VIF analysis is a diagnostic metric used to assess the overall level of multicollinearity in a regression model; it is the largest value of the condition indices calculated from the design (predictors) matrix.[304] [305] [306]

- **Definition:** The condition number is the square root of the ratio of the largest eigenvalue to the smallest eigenvalue of the scaled and centered predictors' matrix $ X $. It quantifies the sensitivity of regression estimates to small changes in the input data.[305] [306]
- **Interpretation:**
  - Condition number < 10: Weak multicollinearity
  - Condition number 10–30: Moderate multicollinearity
  - Condition number > 30: Severe multicollinearity
- **Usage:** Unlike VIF, which measures variance inflation for each predictor individually, the condition number summarizes overall multicollinearity in the entire model. Higher condition numbers mean more unstable estimates and greater risk of unreliable coefficients due to linear dependencies among predictors.[304] [305]

In short, while VIF is per-variable, the condition number is a global summary: a high value signals problematic multicollinearity that may require you to drop, combine, or transform predictor variables.[306] [305] [304]

⁂

```
# <q>maximize Shapiro–Wilk p-value of xnx^nxn (higher p → closer to normal)</q> please e
```

Maximizing the Shapiro–Wilk p-value of x^n means: pick the exponent n that makes the transformed data x^n look most like a sample from a normal distribution, according to the Shapiro–Wilk normality test.[312] [313] [314]

What the Shapiro–Wilk p-value tells you

- The Shapiro–Wilk test checks the null hypothesis "the data come from a normal distribution."[313] [314] [312]
- It returns a p-value:
  - High p-value (e.g. > 0.05): data are compatible with normality (no strong evidence against normal).
  - Low p-value (e.g. < 0.05): data deviate from normality.[315] [313]

So if you apply the test to the transformed data x^n as `shapiro.test(x^n)$p.value`, a **larger p-value means that particular power n makes the distribution of x^n closer to normal** (in the sense of this test).[314] [312] [313]

Turning this into an optimization over n

You can treat n as a tuning parameter:

- For each candidate n (say from -2 to 2), compute x^n.
- Run the Shapiro–Wilk test on x^n and record the p-value.
- Choose n_{opt} as the value of n that gives the **largest** p-value.

In R:

```
normality_score_sw <- function(x, n) {
  x_t <- x ^ n
  x_t <- x_t[is.finite(x_t)]
  if (length(unique(x_t)) < 3) return(NA_real_)
  shapiro.test(x_t)$p.value
}

find_n_shapiro <- function(x, n_grid = seq(-2, 2, by = 0.05)) {
  scores <- sapply(n_grid, function(n) normality_score_sw(x, n))
  valid  <- !is.na(scores)
  best_i <- which.max(scores[valid])
  list(
    n_opt  = n_grid[valid][best_i],
    p_opt  = scores[valid][best_i],
    grid   = n_grid,
    scores = scores
  )
}
```

Here, `scores[k]` is the Shapiro–Wilk p-value for exponent `n_grid[k]`; maximizing it finds the exponent that makes the transformed data "most normal" by that criterion.[313] [314]

Why this is useful

- Many methods (e.g. linear models, t-tests, ANOVA) assume normality of residuals or variables; transforming a skewed variable can help meet this assumption.[316] [317] [313]
- Using the Shapiro–Wilk p-value as an objective gives a **data-driven way** to pick the power n instead of guessing, similar in spirit to how Box–Cox methods choose a λ to improve normality.[318] [319] [320]

⁂

Grid search over n

Grid search over n means: define a grid of candidate exponents, apply each to x, evaluate how normal x^n looks (via Shapiro–Wilk p-value), and pick the n that gives the best score.[337] [338] [339]

Here is a compact, ready-to-use R implementation.

1) Helper: Shapiro–Wilk score for one n

```
normality_score_sw <- function(x, n) {
  x_t <- x ^ n
  x_t <- x_t[is.finite(x_t)]
  if (length(unique(x_t)) < 3) return(NA_real_)  # avoid all-identical errors
  shapiro.test(x_t)$p.value
}
```

2) Grid search function

```
find_n_shapiro <- function(x,
                           n_grid = seq(-2, 2, by = 0.05)) {
  scores <- sapply(n_grid, function(n) normality_score_sw(x, n))
  valid  <- !is.na(scores)
  if (!any(valid)) stop("No valid n in grid (all transforms constant/invalid).")
  best_i <- which.max(scores[valid])
  list(
    n_opt  = n_grid[valid][best_i],  # exponent with max p-value
    p_opt  = scores[valid][best_i],  # corresponding p-value
    grid   = n_grid,
    scores = scores
  )
}
```

3) How to use it

```
res <- find_n_shapiro(x)     # x is your numeric vector
res$n_opt                    # optimal n
res$p_opt                    # Shapiro–Wilk p-value at n_opt
x_trans <- x ^ res$n_opt     # transformed variable
```

- You can change the search range or step, e.g. `seq(-1, 1, by = 0.01)`, if your data require gentler powers.[340] [341]
- The idea is: the **grid** is all candidate n, the **score** is the Shapiro p-value for x^n, and you choose the n that maximizes that score to make x^n as close to normal as this test can detect.[338] [339] [337]

⁂

1. https://en.wikipedia.org/wiki/Ordered_weighted_averaging
2. http://akira.ruc.dk/~bulskov/undervisning/E2003/OWA_yager.pdf
3. https://www.sciencedirect.com/topics/computer-science/ordered-weighted-averaging
4. https://ui.adsabs.harvard.edu/abs/2019IJGS...48..843D/abstract
5. https://doaj.org/article/51f7358ab2ad4536824eb697ad43dcb6
6. https://www.ia.pw.edu.pl/~wogrycza/publikacje/artykuly/mylncs12.pdf
7. https://www.kybernetika.cz/content/2015/4/712/paper.pdf
8. https://support.sas.com/resources/papers/proceedings/pdfs/sgf2008/202-2008.pdf
9. https://sinbad2.ujaen.es/sites/default/files/publications/2023- BUI OWA.pdf
10. https://onlinelibrary.wiley.com/doi/full/10.1002/int.22168
11. https://www.sciencedirect.com/science/article/abs/pii/S0888613X23001378
12. https://ieeexplore.ieee.org/document/6601708/
13. https://onlinelibrary.wiley.com/doi/full/10.1002/int.22376
14. https://www.tandfonline.com/doi/abs/10.1080/03081079.2019.1668386
15. https://www.sciencedirect.com/science/article/abs/pii/S016501140200369X
16. https://www.atlantis-press.com/proceedings/ifsa-eusflat-agop-21/125960368
17. https://www.semanticscholar.org/paper/Orness-Measure-of-OWA-Operators:-A-New-Approach-Kishor-Singh/278d41c7c877759154d9fc501906ec9bace9852a
18. https://academica-e.unavarra.es/bitstreams/8e2042e7-2f09-4870-98e0-8aeae84e4d83/download
19. https://academica-e.unavarra.es/bitstreams/c3555ce4-12e4-4d71-936e-6dee9499428b/download
20. https://www.diva-portal.org/smash/get/diva2:1857945/FULLTEXT02.pdf
21. https://www.kybernetika.cz/content/2015/4/712/paper.pdf
22. https://en.wikipedia.org/wiki/Ordered_weighted_averaging
23. https://support.sas.com/resources/papers/proceedings/pdfs/sgf2008/202-2008.pdf
24. https://academica-e.unavarra.es/bitstreams/c3555ce4-12e4-4d71-936e-6dee9499428b/download
25. https://ui.adsabs.harvard.edu/abs/2019IJGS...48..843D/abstract

26. https://arxiv.org/pdf/1801.10545.pdf
27. https://www.sciencedirect.com/science/article/abs/pii/S0888613X23001378
28. https://annals-csis.org/Volume_29/drp/pdf/6.pdf
29. https://pearl.plymouth.ac.uk/cgi/viewcontent.cgi?article=1500&context=nm-research
30. https://onlinelibrary.wiley.com/doi/full/10.1002/int.22376
31. https://www.slideshare.net/slideshow/owa-method-206236893/206236893
32. https://www.sciencedirect.com/science/article/pii/S0307904X19301271
33. https://www.academia.edu/128076583/Orness_Measure_of_OWA_Operators_A_New_Approach
34. https://www.sciencedirect.com/science/article/pii/S0165011420303419
35. https://www.scribd.com/document/554812712/yager1988
36. https://www.sciencedirect.com/science/article/pii/S0165011408003321
37. https://diposit.ub.edu/dspace/bitstream/2445/129884/1/679648.pdf
38. https://www.ia.pw.edu.pl/~wogrycza/publikacje/artykuly/mylncs12.pdf
39. https://www.sciencedirect.com/science/article/pii/S0020025521011014
40. https://www.scitepress.org/papers/2008/14835/14835.pdf
41. https://ieeexplore.ieee.org/iel2/632/7149/00287131.pdf
42. https://en.wikipedia.org/wiki/Ordered_weighted_averaging
43. https://www.kybernetika.cz/content/2015/4/712/paper.pdf
44. https://nrc-publications.canada.ca/eng/view/accepted/?id=62a4761a-b73b-4e0d-a434-acf5f20eaa46
45. https://arxiv.org/pdf/1801.10545.pdf
46. https://support.sas.com/resources/papers/proceedings/pdfs/sgf2008/202-2008.pdf
47. https://ui.adsabs.harvard.edu/abs/2019IJGS...48..843D/abstract
48. https://annals-csis.org/Volume_29/drp/pdf/6.pdf
49. https://pearl.plymouth.ac.uk/cgi/viewcontent.cgi?article=1500&context=nm-research
50. https://www.ca3-uninova.org/docs/2003-BJ-EJOR.pdf
51. https://www.academia.edu/986036/On_the_issue_of_obtaining_OWA_operator_weights
52. https://www.iosrjournals.org/iosr-jce/papers/Vol16-issue3/Version-4/J016346574.pdf
53. https://www.diva-portal.org/smash/get/diva2:1857945/FULLTEXT02.pdf
54. https://ir.lib.nycu.edu.tw/bitstream/11536/12331/1/000235502300002.pdf
55. https://www.sciencedirect.com/science/article/pii/S0165011424000095
56. https://onlinelibrary.wiley.com/doi/full/10.1002/int.22376
57. https://www.sciencedirect.com/science/article/pii/S0165011408003321
58. https://www.sciencedirect.com/science/article/pii/S0307904X19301271
59. http://ieeexplore.ieee.org/document/4630589/
60. https://www.academia.edu/47597639/Orness_For_Idempotent_Aggregation_Functions
61. https://diposit.ub.edu/dspace/bitstream/2445/129884/1/679648.pdf
62. https://www.tandfonline.com/doi/full/10.1080/19475683.2014.904439
63. https://www.sciencedirect.com/science/article/pii/S0020025521011014

64. https://pmc.ncbi.nlm.nih.gov/articles/PMC4127207/
65. https://www.sciencedirect.com/science/article/pii/S0165011423003226
66. https://www.w3schools.com/r/r_data_types.asp
67. https://www.programiz.com/r/data-types
68. https://www.index.dev/blog/check-data-types-in-r-programming
69. https://swcarpentry.github.io/r-novice-inflammation/13-supp-data-structures.html
70. https://www.datacamp.com/doc/r/datatypes
71. https://www.tutorialspoint.com/r/r_data_types.htm
72. https://www.geeksforgeeks.org/r-language/r-data-types/
73. https://www.geeksforgeeks.org/r-language/classification-in-r-programming/
74. https://www.coursera.org/articles/r-data-types
75. https://www.c-sharpcorner.com/article/matrix-in-r-naming-cloumn-using-colames-function-and-accessing-matrix-elemen/
76. https://www.dummies.com/article/technology/programming-web-design/r/how-to-name-matrix-rows-and-columns-in-r-141615/
77. https://ieeexplore.ieee.org/document/6601708/
78. https://www.geeksforgeeks.org/r-language/change-column-name-of-a-given-dataframe-in-r/
79. https://community.fabric.microsoft.com/t5/Desktop/rename-a-column-header-in-matrix-visual-when-column-represents/m-p/3077461
80. https://stackoverflow.com/questions/69136007/changing-column-names-of-a-matrix-in-r-without-using-colnames
81. https://www.projectpro.io/recipes/change-column-headers-of-dataframe-r
82. https://www.datanovia.com/en/lessons/rename-data-frame-columns-in-r/
83. https://community.powerbi.com/t5/Desktop/How-to-change-matrix-column-header-label/td-p/2760511
84. https://campus.datacamp.com/courses/free-introduction-to-r/chapter-3-matrices-3?ex=3
85. https://www.youtube.com/watch?v=yL3GfwSp0t4
86. https://www.geeksforgeeks.org/r-language/how-to-create-a-histogram-of-two-variables-in-r/
87. https://stackoverflow.com/questions/33502027/how-to-make-a-histogram-from-a-matrix-in-r
88. https://r-graph-gallery.com/histogram_several_group.html
89. https://www.r-bloggers.com/2023/08/how-to-plot-multiple-histograms-with-base-r-and-ggplot2/
90. https://bookdown.org/dli/rguide/histogram.html
91. https://www.rdocumentation.org/packages/psych/versions/2.5.6/topics/multi.hist
92. https://www.reddit.com/r/rstats/comments/myi5h9/q_how_to_use_facet_wraps_in_ggplot2_to_create/
93. https://www.tutorialspoint.com/how-to-plot-matrix-columns-as-lines-in-base-r
94. https://bookdown.org/rwnahhas/IntroToR/continuous-histogram.html
95. https://www.sthda.com/english/wiki/ggplot2-histogram-plot-quick-start-guide-r-software-and-data-visualization
96. https://www.sfu.ca/~mjbrydon/tutorials/BAinR/visualize.html
97. https://www.spsanderson.com/steveondata/posts/2023-09-15/index.html
98. https://www.statology.org/histogram-two-variables-in-r/

99. https://www.r-bloggers.com/2023/09/histograms-with-two-or-more-variables-in-r/
100. https://stackoverflow.com/questions/19268858/histogram-for-multiple-variables-in-r
101. https://www.datacamp.com/tutorial/make-histogram-basic-r
102. https://www.geeksforgeeks.org/data-visualization/working-with-multiple-plots-faceting/
103. https://www.geeksforgeeks.org/r-language/histograms-in-r-language/
104. https://www.datamentor.io/r-programming/histogram
105. https://stackoverflow.com/questions/56036577/how-to-plot-multiple-facets-histogram-with-ggplot-in-r
106. https://campus.datacamp.com/courses/free-introduction-to-r/chapter-3-matrices-3?ex=8
107. https://www.spsanderson.com/steveondata/posts/2024-05-08/
108. https://adv-r.hadley.nz/subsetting.html
109. https://www.tutorialspoint.com/how-to-select-a-column-of-a-matrix-by-column-name-in-r
110. https://www.datacamp.com/tutorial/make-histogram-basic-r
111. https://www.r-bloggers.com/2023/08/how-to-plot-multiple-histograms-with-base-r-and-ggplot2/
112. https://stackoverflow.com/questions/5744694/extract-matrix-column-values-by-matrix-column-name
113. https://www.geeksforgeeks.org/r-language/how-to-select-specific-columns-in-r-dataframe/
114. https://www.anyamemensah.com/blog/select-col-name
115. https://benwhalley.github.io/just-enough-r/selecting-columns.html
116. https://www.youtube.com/watch?v=Rg4U_Z9BBRo
117. https://en.wikipedia.org/wiki/Quasi-arithmetic_mean
118. https://en.wikipedia.org/wiki/Weighted_arithmetic_mean
119. https://files.ele-math.com/articles/mia-22-77.pdf
120. https://www.cuemath.com/weighted-mean-formula/
121. https://allen.in/jee/maths/weighted-arithmetic-mean
122. https://byjus.com/jee/arithmetic-mean/
123. https://eudml.org/doc/280717
124. https://publi.math.unideb.hu/load_doc.php?p=1266&t=pap
125. https://en.wikipedia.org/wiki/Quasi-arithmetic_mean
126. https://en.wikipedia.org/wiki/Weighted_arithmetic_mean
127. https://files.ele-math.com/articles/mia-22-77.pdf
128. https://dl.acm.org/doi/10.1007/978-3-642-41550-0_3
129. https://www.sciencedirect.com/science/article/pii/S0047259X09000475
130. https://allen.in/jee/maths/weighted-arithmetic-mean
131. https://arxiv.org/pdf/2301.10980.pdf
132. https://www.datanovia.com/en/lessons/select-data-frame-columns-in-r/
133. https://www.dataquest.io/blog/add-column-to-dataframe-in-r/
134. https://stackoverflow.com/questions/10085806/extracting-specific-columns-from-a-data-frame
135. https://www.geeksforgeeks.org/r-language/filter-dataframe-columns-in-r-by-given-condition/
136. https://www.datacamp.com/tutorial/subsets-in-r

137. https://www.geeksforgeeks.org/r-language/how-to-select-dataframe-columns-by-index-in-r/
138. https://benwhalley.github.io/just-enough-r/selecting-columns.html
139. https://www.anyamemensah.com/blog/select-col-name
140. https://www.w3schools.com/r/r_data_frames.asp
141. https://dplyr.tidyverse.org/reference/select.html
142. https://www.datanovia.com/en/lessons/select-data-frame-columns-in-r/
143. https://www.dataquest.io/blog/add-column-to-dataframe-in-r/
144. https://stackoverflow.com/questions/10085806/extracting-specific-columns-from-a-data-frame
145. https://www.geeksforgeeks.org/r-language/filter-dataframe-columns-in-r-by-given-condition/
146. https://www.datacamp.com/tutorial/subsets-in-r
147. https://www.geeksforgeeks.org/r-language/how-to-select-dataframe-columns-by-index-in-r/
148. https://benwhalley.github.io/just-enough-r/selecting-columns.html
149. https://www.anyamemensah.com/blog/select-col-name
150. https://www.w3schools.com/r/r_data_frames.asp
151. https://dplyr.tidyverse.org/reference/select.html
152. https://www.sthda.com/english/wiki/ggplot2-histogram-plot-quick-start-guide-r-software-and-data-visualization
153. https://www.datacamp.com/tutorial/make-histogram-basic-r
154. https://www.geeksforgeeks.org/data-visualization/working-with-multiple-plots-faceting/
155. https://bookdown.org/dli/rguide/histogram.html
156. https://r-graph-gallery.com/histogram_several_group.html
157. https://www.datamentor.io/r-programming/histogram
158. AggWaFit718.R
159. 978-3-319-46762-7.pdf
160. AggWaFit718.R
161. KeiHotels.txt
162. 978-3-319-46762-7.pdf
163. AggWaFit718.R
164. KeiHotels.txt
165. AggWaFit718.R
166. 978-3-319-46762-7.pdf
167. AggWaFit718.R
168. 978-3-319-46762-7.pdf
169. KeiHotels.txt
170. image.jpg
171. image.jpg
172. AggWaFit718.R
173. KeiHotels.txt
174. 978-3-319-46762-7.pdf

175. AggWaFit718.R
176. 978-3-319-46762-7.pdf
177. KeiHotels.txt
178. AggWaFit718.R
179. KeiHotels.txt
180. 978-3-319-46762-7.pdf
181. https://www.sciencedirect.com/science/article/pii/0165011489901942
182. https://sal.aalto.fi/publications/pdf-files/tkan14b_public.pdf
183. AggWaFit718.R
184. 978-3-319-46762-7.pdf
185. https://www.sciencedirect.com/science/article/pii/0022247X9190213J
186. KeiHotels.txt
187. https://ieeexplore.ieee.org/document/873585/
188. https://www.worldscientific.com/doi/full/10.1142/S0218488599000507
189. https://onlinelibrary.wiley.com/doi/10.1155/2015/315340
190. https://guides.rostools.org/posit-cloud
191. https://help.lafayette.edu/posit/
192. https://posit.cloud
193. https://utdata.github.io/rwdir/posit-cloud.html
194. https://docs.posit.co/cloud/guide/data/
195. https://www.databricks.com/blog/2018/07/06/sharing-r-notebooks-using-rmarkdown.html
196. https://www.youtube.com/watch?v=UWn1TYXEC7o
197. https://semo-gabd.github.io/notes/notes03.html
198. https://www.youtube.com/watch?v=GEvyAqYa8p0
199. https://www.datacamp.com/blog/jupyter-and-r-markdown-notebooks-with-r
200. https://docs.posit.co/connect/user/rmarkdown/
201. https://www.youtube.com/watch?v=2oOvcs8XuSs
202. https://docs.posit.co/connect-cloud/user/content/rmarkdown.html
203. https://www.appsilon.com/post/deploying-to-rstudio-connect
204. https://github.com/att/rcloud.rmd
205. https://docs.posit.co/connect/user/publishing-rstudio/
206. https://support.researchallofus.org/hc/en-us/articles/22078658566804-Using-RStudio-on-the-Researcher-Workbench
207. https://www.reddit.com/r/Rlanguage/comments/1clfpng/help_with_uploading_csv_files_onto_my_r_cloud/
208. https://forum.posit.co/t/uploading-files-onto-rstudio/176843
209. https://posit.cloud/learn/recipes
210. https://forum.posit.co/t/how-to-upload-classical-rstudio-file-into-rstudio-cloud/6182
211. https://guides.rostools.org/posit-cloud

212. https://help.lafayette.edu/posit/
213. https://utdata.github.io/rwdir/posit-cloud.html
214. https://posit.cloud
215. https://support.posit.co/hc/en-us/articles/200713893-Uploading-and-Downloading-Files-in-RStudio-Workbench-or-RStudio-Server
216. https://docs.posit.co/cloud/guide/data/
217. https://www.youtube.com/watch?v=GEvyAqYa8p0
218. https://rmarkdown.rstudio.com/lesson-2.html
219. https://semo-gabd.github.io/notes/notes03.html
220. https://docs.posit.co/connect-cloud/user/content/rmarkdown.html
221. https://www.youtube.com/watch?v=UWn1TYXEC7o
222. https://bio304-class.github.io/bio304-fall2017/rstudio-cloud-uploading.html
223. https://www.youtube.com/watch?v=2oOvcs8XuSs
224. https://www.youtube.com/watch?v=FSiDrlOAjgo
225. https://support.researchallofus.org/hc/en-us/articles/22078658566804-Using-RStudio-on-the-Researcher-Workbench
226. https://www.youtube.com/watch?v=55ZZZhpt0vY
227. https://stackoverflow.com/questions/65305729/open-r-markdown-in-viewer-pane
228. https://docs.posit.co/connect/user/rmarkdown/
229. https://www.reddit.com/r/RStudio/comments/1dsvfsk/no_option_to_upload_files/
230. https://lpsolve.sourceforge.net/5.5/
231. https://cran.r-project.org/view=Optimization
232. https://pmc.ncbi.nlm.nih.gov/articles/PMC12403259/
233. https://www.6sigma.us/six-sigma-in-focus/normality-test-lean-six-sigma/
234. http://www.pelagicos.net/MARS4910/workshops/MARS4910_2020_Statistics_Normality.pdf
235. https://www.statisticshowto.com/probability-and-statistics/normal-distributions/box-cox-transformation/
236. https://www.is.uni-freiburg.de/resources/computational-economics/5_OptimizationR.pdf
237. https://stackoverflow.com/questions/26617587/finding-optimal-lambda-for-box-cox-transform-in-r
238. https://www.r-bloggers.com/2022/10/box-cox-transformation-in-r/
239. https://www.rdocumentation.org/packages/forecast/versions/8.24.0/topics/BoxCox.lambda
240. https://r-coder.com/box-cox-transformation-r/
241. https://petersonr.github.io/bestNormalize/reference/boxcox.html
242. https://cran.r-project.org/web/packages/AID/AID.pdf
243. https://www.sciencedirect.com/science/article/pii/S2213398422000859
244. https://www.youtube.com/watch?v=AJw91UVYNsg
245. https://www.sciencedirect.com/science/article/abs/pii/S0377042723005939
246. https://www.kdnuggets.com/2018/05/optimization-using-r.html

247. https://stackoverflow.com/questions/46779740/set-up-linear-programming-optimization-in-r-using-lpsolve
248. https://www.6sigma.us/six-sigma-in-focus/box-cox-transformation/
249. https://files.eric.ed.gov/fulltext/EJ1259038.pdf
250. https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/shapiro.test
251. https://www.sthda.com/english/wiki/normality-test-in-r
252. https://stackoverflow.com/questions/25224318/how-to-apply-shapiro-test-in-r
253. https://www.geeksforgeeks.org/r-language/shapiro-wilk-test-in-r-programming/
254. https://www.rdocumentation.org/packages/rstatix/versions/0.7.3/topics/shapiro_test
255. https://rdrr.io/cran/rstatix/man/shapiro_test.html
256. https://r-packages.io/packages/psyntur/shapiro_test
257. https://search.r-project.org/CRAN/refmans/rstatix/html/shapiro_test.html
258. https://rfaqs.com/analysis/assumptions/shapiro-wilk-test-in-r/
259. https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/shapiro.test
260. https://stackoverflow.com/questions/25224318/how-to-apply-shapiro-test-in-r
261. http://www.pelagicos.net/MARS4910/workshops/MARS4910_2020_Statistics_Normality.pdf
262. https://www.statisticshowto.com/probability-and-statistics/normal-distributions/box-cox-transformation/
263. powermean-stats3.txt
264. WAM-stats3.txt
265. OWA-stats3.txt
266. choquet-stats3.txt
267. powermean-stats3.txt
268. WAM-stats3.txt
269. OWA-stats3.txt
270. powermean-stats31.txt
271. powermean-stats3.txt
272. WAM-stats3.txt
273. OWA-stats3.txt
274. choquet-stats3.txt
275. powermean-stats23.txt
276. powermean-stats3.txt
277. WAM-stats3.txt
278. OWA-stats3.txt
279. choquet-stats3.txt
280. powermean-stats31.txt
281. image.jpg
282. powermean-stats3.txt
283. WAM-stats3.txt

284. OWA-stats3.txt
285. choquet-stats3.txt
286. powermean-stats31.txt
287. powermean-stats23.txt
288. https://www.byteplus.com/en/topic/540333
289. https://learn.microsoft.com/en-us/semantic-kernel/concepts/ai-services/chat-completion/chat-history
290. https://www.eesel.ai/blog/chatgpt-response-time
291. https://stackoverflow.com/questions/8655765/chat-server-whats-the-bestoptimized-way-to-save-a-conversation-log
292. https://www.youtube.com/watch?v=-xQve1WKIlo
293. https://www.daily.co/blog/how-to-add-chat-history-to-your-custom-daily-calls-with-only-react-client-side-code/
294. https://community.openai.com/t/chat-history-and-semantic-search/460984
295. https://support.google.com/gemini/thread/361663397/client-chats-with-uploaded-files-input-token-count-is-too-large?hl=en
296. https://bytebytego.com/courses/system-design-interview/design-a-chat-system
297. https://strandsagents.com/latest/documentation/docs/user-guide/concepts/agents/conversation-management/
298. https://community.openai.com/t/enhancing-load-times-for-lengthy-chatgpt-sessions/569142
299. https://www.reddit.com/r/ChatGPTPro/comments/1b11td4/how_to_handle_degrading_performance_in_longer/
300. https://www.companionlink.com/blog/2025/08/why-chatgpt-5-is-glacially-slow-on-long-chats-and-what-to-do-about-it/
301. https://www.youtube.com/watch?v=xc__mwjG0L0
302. https://github.com/open-webui/open-webui/discussions/13787
303. https://quidget.ai/blog/ai-automation/10-tips-to-reduce-chatbot-response-time/
304. https://pmc.ncbi.nlm.nih.gov/articles/PMC6900425/
305. https://fiveable.me/linear-modeling-theory-and-applications/unit-16/detecting-multicollinearity-vif-condition-number/study-guide/MiDWzuI9sLqO8Ri8
306. https://faculty.insead.edu/hubert-gatignon/documents/HG Notes-Identification of Multicollinearity-VIF and Conditioning Number_20140304.pdf
307. https://www.investopedia.com/terms/v/variance-inflation-factor.asp
308. https://en.wikipedia.org/wiki/Variance_inflation_factor
309. https://corporatefinanceinstitute.com/resources/data-science/variance-inflation-factor-vif/
310. https://www.statlect.com/fundamentals-of-statistics/multicollinearity
311. https://www.sciencedirect.com/topics/mathematics/variance-inflation-factor
312. https://en.wikipedia.org/wiki/Shapiro–Wilk_test
313. https://builtin.com/data-science/shapiro-wilk-test
314. https://statisticsbyjim.com/glossary/shapiro-wilk-normality-test/
315. https://www.atlas.org/spaces/solve/understanding-p-value-interpretation-q9C7pZkZdEJfKYnKKzddLr

316. https://www.6sigma.us/six-sigma-in-focus/normality-test-lean-six-sigma/
317. http://www.pelagicos.net/MARS4910/workshops/MARS4910_2020_Statistics_Normality.pdf
318. https://www.r-bloggers.com/2022/10/box-cox-transformation-in-r/
319. https://cran.r-project.org/web/packages/AID/AID.pdf
320. https://www.statisticshowto.com/probability-and-statistics/normal-distributions/box-cox-transformation/
321. https://community.jmp.com/t5/JMP-Knowledge-Base/How-do-I-interpret-the-Shapiro-Wilk-test-for-normality-in-JMP/ta-p/575365
322. https://www.reddit.com/r/statistics/comments/1e27kt5/q_my_data_looks_very_normally_distributed_but/
323. https://support.safe.com/hc/en-us/articles/25407402335629-Perform-a-Shapiro-Wilk-Statistical-Test-using-R-or-Python
324. https://www.reddit.com/r/AskStatistics/comments/1e27hct/this_look_normally_distributed_but_shapirowilk/
325. https://www.tandfonline.com/doi/abs/10.1080/00949655.2019.1658763
326. https://www.spss-tutorials.com/spss-shapiro-wilk-test-for-normality/
327. https://arxiv.org/html/2501.14845v1
328. https://variation.com/wp-content/distribution_analyzer_help/hs141.htm
329. https://www.sciencedirect.com/topics/mathematics/wilk-test
330. https://www.stata.com/manuals/rswilk.pdf
331. https://www.geeksforgeeks.org/r-language/shapiro-wilk-test-in-r-programming/
332. https://www.reddit.com/r/AskStatistics/comments/cxsy51/kolmogorov_vs_shapirowilk/
333. https://www.statskingdom.com/doc_shapiro_wilk.html
334. https://www.youtube.com/watch?v=iNrBDxMZWxg
335. https://www.youtube.com/watch?v=9KKAX5av7S4
336. https://www.dmaic.com/faq/shapiro-wilk-test/
337. https://en.wikipedia.org/wiki/Shapiro–Wilk_test
338. https://builtin.com/data-science/shapiro-wilk-test
339. https://statisticsbyjim.com/glossary/shapiro-wilk-normality-test/
340. http://www.pelagicos.net/MARS4910/workshops/MARS4910_2020_Statistics_Normality.pdf
341. https://www.statisticshowto.com/probability-and-statistics/normal-distributions/box-cox-transformation/