ChatGPT

Day Strain Calculation from Apple Health Data (TypeScript Implementation)

Figure: The WHOOP Strain scale ranges from 0 (no exertion) to 21 (maximal exertion), categorized into Light (0–9), Moderate (10–13), High (14–17), and All Out (18–21) intensity levels [1]. This "strain" metric is a combined measure of cardiovascular and muscular exertion, quantifying the total stress on the body (physical *and* even mental) for the day [2]. A higher strain indicates greater cumulative load, and **the scale is logarithmic** – meaning as strain rises, it becomes progressively harder to increase it further [3] [4]. In other words, moving from strain 18 to 19 represents much more work than going from 8 to 9.

Cardiovascular Load – Heart Rate Zones Contribution

To calculate the cardiovascular portion of Day Strain, we use heart rate data from HealthKit, distributed across intensity *zones*. These zones are determined by your heart rate relative to your resting and maximum heart rate. We define five zones (as commonly used in training literature and by WHOOP):

- **Zone 1:** ~50–60% of max HR (low intensity)
- **Zone 2:** ~60–70% of max HR (moderate aerobic activity)
- **Zone 3:** ~70–80% of max HR (vigorous aerobic activity)
- **Zone 4:** ~80–90% of max HR (high intensity)
- **Zone 5:** ~90–100% of max HR (near maximal effort)

(Note: One can calculate max HR as `220 - age` *(Fox formula) or use a value from HealthKit/user profile* [5]*. Resting HR can be read from HealthKit's* `restingHeartRate` *if available.)*

Time in Zone: Using the heart rate samples throughout the day, we determine how long the heart spent in each zone. WHOOP's strain algorithm emphasizes **time spent in higher heart rate zones** – the higher the HR zone, the more strain accumulates per minute [6]. For example, **30 minutes at 80% max HR contributes much more strain than 30 minutes at 50%** [6]. We implement this by assigning a weight to each zone, increasing with intensity. A common heuristic (inspired by research and WHOOP's hints) is:

- Zone 1 = 1 point per minute
- Zone 2 = 2 points per minute
- Zone 3 = 3 points per minute
- Zone 4 = 4 points per minute
- Zone 5 = 5 points per minute

Using these weights, we compute a **cardio load score** as the weighted sum of minutes in each zone. For example, if a user spent 10 min in Zone 3, 20 min in Zone 4, and 5 min in Zone 5, the weighted sum would be `10×3 + 20×4 + 5×5 = 135` points [7] [8]. This raw score will later be scaled logarithmically to contribute to the 0–21 strain.

Muscular Load – Strength Training Contribution

Cardiovascular load alone doesn't capture strain from **strength/resistance training** or other muscular work. WHOOP therefore also factors in a **muscular load**, which accounts for the mechanical work your muscles perform [9]. In WHOOP's system, muscular load is derived from accelerometer/gyroscope data and logged workouts, incorporating both **volume** (mass moved) and **intensity** (effort, speed of movement) [10] [11]. Essentially:

- *Volume* = effective mass moved (taking into account bodyweight and which muscles are involved – e.g. a squat moves more mass than a bicep curl) [12] [13].
- *Intensity* = effort level, including movement speed and proximity to failure (lifting heavy weight or explosive reps increases intensity) [14] [15].

In our HealthKit-based implementation, we may not have detailed sensor data for intensity, so we use a **heuristic based on workout data**. We can query for strength training workouts (e.g. HealthKit workout type "Functional Strength Training" or similar) and use whatever metrics are available as a proxy for muscular load. For instance, if the user logs sets and reps with weight (via a connected app), we can sum up a "volume" metric:

$$**muscleVolume** = \sum_{\text{exercises}} (\text{weight} \times \text{reps} \times \text{sets})$$

This aligns with WHOOP's description that muscular strain could be derived from the sum of weight×reps×sets (with some normalization) [4]. If detailed lift data isn't available, an approximation could be made using the **Active Energy** or **total calories** burned during strength workouts (as higher energy expenditure in a strength session generally means more volume or intensity). We assume that HealthKit provides either the detailed metrics (through workout metadata or additional tracking apps) or we use the total energy/duration of strength workouts as a rough stand-in for muscular load.

From the muscular data, we compute a **muscle load score** similar to the cardio points. For example, if a user lifted a total volume of say 10,000 kg (aggregated across all exercises in the day), that number (with appropriate scaling) would feed into the strain calculation. We might take the sum of weight·reps across all sets in the day as a base muscular load number.

Combining Cardiovascular and Muscular Load into "Day Strain"

The Day Strain score (0–21) is produced by combining the cardiovascular and muscular loads and applying a **logarithmic scaling**. The use of a log function reflects diminishing returns: as total workload increases, each additional unit of work contributes less to the strain score, preventing the number from exceeding 21 [4]. In practice, WHOOP uses a proprietary formula, but it has been described conceptually as:

- **Cardio Strain Component** ≈ $\log_a$(*weighted cardio points*)
- **Muscular Strain Component** ≈ $\log_a$(*muscle volume points*)

…and these are integrated (with some normalization factors and a baseline) into one strain score [4]. We will implement a simplified version: we add the cardio and muscle point totals into a single *total load*, then apply a logarithmic scaling to that sum. A small constant may be added before taking log, and a scaling factor applied after, to calibrate the result into the 0–21 range. Finally, we cap the value at 21.

This approach ensures that spending a long time in very high heart rate zones (cardio load) or performing a lot of heavy lifting (muscular load) will increase strain, but extraordinarily large loads will yield incrementally smaller increases in the strain score (approaching an asymptotic limit around 21). For example, doubling your total workload in a day might not double your strain – it might raise it from, say, 15 to 18, due to the logarithmic compression.

Implementation in TypeScript (using @kingstinct/react-native-healthkit)

Below is a TypeScript function that calculates the Day Strain given a date, using Apple HealthKit data accessed via the `@kingstinct/react-native-healthkit` library. It queries heart rate samples and strength workouts from HealthKit, computes cardiovascular and muscular load, and then combines them into the final strain score on a 0–21 scale. The code follows the structure and style of the Kingstinct HealthKit API (using `queryQuantitySamples` for heart rate, etc., as in the provided example):

```
import HealthKit, {
  HKQuantityTypeIdentifier,
  HKWorkoutActivityType,
  HKUnit,
  HKQuantitySample,
  HKWorkout
} from "@kingstinct/react-native-healthkit";

/**
 * Calculate the "Day Strain" score (0–21) for a given day using HealthKit data.
 * Combines cardiovascular load (from heart rate zones) and muscular load (from
strength workouts).
 */
export async function calculateDayStrain(date: Date): Promise<number> {
  // 1. Define the time range for the day (midnight to 23:59 of the given date)
  const startOfDay = new Date(date.getFullYear(), date.getMonth(),
date.getDate());
  const endOfDay = new Date(startOfDay.getTime() + 24 * 60 * 60 *
1000); // next midnight

  // 2. Query heart rate samples for the day
  const heartRateSamples: HKQuantitySample[] = await
HealthKit.queryQuantitySamples(
    HKQuantityTypeIdentifier.heartRate,
    {
      unit: "count/min",               // bpm unit for heart rate
      from: startOfDay.toISOString(),
      to: endOfDay.toISOString()
    }
  );

  // 3. Determine user’s resting and max HR for zone calculations
  // Try to get resting HR from HealthKit (e.g., lowest heart rate of day or
dedicated RestingHR sample)
  let restingHR = 60;
  try {
    const restingSample = await
HealthKit.getMostRecentQuantitySample(HKQuantityTypeIdentifier.restingHeartRate);
    if (restingSample && restingSample.quantity) {
      restingHR = restingSample.quantity;  // assuming quantity is number in bpm
    }
  } catch {}
  // Estimate max HR (could be user-specific or use formula 220-age if DOB
available)
  let maxHR = 190;
  try {
    const birthDate = await HealthKit.getDateOfBirth();
    if (birthDate) {
      const age = new Date().getFullYear() - new Date(birthDate).getFullYear();
      maxHR = 220 - age;  // simple Fox formula estimate
    }
  } catch {}
  // Ensure maxHR is at least higher than any observed HR
  heartRateSamples.forEach(sample => {
    if (sample.quantity && sample.quantity > maxHR) {
      maxHR = sample.quantity;
    }
  });

  // 4. Define heart rate zone thresholds using Heart Rate Reserve (HRR) method
  const HRR = maxHR - restingHR;
  const zones = [
    restingHR + 0.5 * HRR,  // threshold between Zone1 and Zone2 (50% HRR)
    restingHR + 0.6 * HRR,  // Zone2 threshold (60% HRR)
    restingHR + 0.7 * HRR,  // Zone3 threshold (70% HRR)
    restingHR + 0.8 * HRR,  // Zone4 threshold (80% HRR)
    restingHR + 0.9 * HRR   // Zone5 threshold (90% HRR) – above this is zone5
  ];

  // 5. Calculate time in each zone from heart rate samples
  // Initialize zone duration counters (in minutes)
  let zoneMinutes = [0, 0, 0, 0, 0];
  if (heartRateSamples.length > 0) {
    // Sort samples by start time
    heartRateSamples.sort((a, b) => new Date(a.startDate).getTime() - new
Date(b.startDate).getTime());
```

```
    for (let i = 0; i < heartRateSamples.length; i++) {
      const sample = heartRateSamples[i];
      const hr = sample.quantity;  // heart rate value in bpm
      if (typeof hr !== "number") continue;
      // Determine which zone this heart rate falls into
      let zoneIndex = 0;
      if (hr >= zones[4]) {
        zoneIndex = 4;
      } else if (hr >= zones[3]) {
        zoneIndex = 3;
      } else if (hr >= zones[2]) {
        zoneIndex = 2;
      } else if (hr >= zones[1]) {
        zoneIndex = 1;
      } else if (hr >= zones[0]) {
        zoneIndex = 0;
      } else {
        // hr below 50% HRR (below zones[0]) counts as Zone0 (very light) – we
treat as Zone 0 here (no strain)
        zoneIndex = -1;
      }

      if (zoneIndex >= 0) {
        // Calculate duration until next sample or end of day
        let endTime = endOfDay;
        if (i < heartRateSamples.length - 1) {
          endTime = new Date(heartRateSamples[i + 1].startDate);
        }
        const startTime = new Date(sample.startDate);
        let durationMin = (endTime.getTime() - startTime.getTime()) / (1000 *
60);
        if (durationMin < 0) durationMin = 0;
        zoneMinutes[zoneIndex] += durationMin;
      }
    }
  }

  // 6. Compute cardiovascular strain points using weighted zone durations
  const zoneWeights = [1, 2, 3, 4, 5];  // weight multipliers for zones 1–5
  let cardioPoints = 0;
  for (let z = 0; z < zoneMinutes.length; z++) {
    cardioPoints += zoneMinutes[z] * zoneWeights[z];
  }

  // 7. Query strength-type workouts for the day to assess muscular load
  const allWorkouts: HKWorkout[] = await HealthKit.queryWorkouts({
    // (the API might allow specifying date range; if not, we filter below)
    energyUnit: "kcal"
```

~~~
  });
  // Filter workouts that fall in the day range and are strength training types
  const muscleWorkouts = allWorkouts.filter(w => {
    const wStart = new Date(w.startDate);
    return wStart >= startOfDay && wStart < endOfDay && (
      w.workoutActivityType ===
HKWorkoutActivityType.functionalStrengthTraining ||
      w.workoutActivityType ===
HKWorkoutActivityType.traditionalStrengthTraining ||
      w.workoutActivityType === HKWorkoutActivityType.crossTraining
    );
  });

  // 8. Compute muscular load points
  let musclePoints = 0;
  for (const workout of muscleWorkouts) {
    // If detailed weight/rep data is available in metadata, use it
    if (workout.metadata && workout.metadata.TotalWeightLifted) {
      // Suppose an app saved total weight lifted (kg) in metadata
      musclePoints += workout.metadata.TotalWeightLifted;
    } else if (workout.totalEnergyBurned) {
      // Fallback: use active energy (kcal) as a rough proxy for muscular work
      const energy = typeof workout.totalEnergyBurned === "object"
        ? workout.totalEnergyBurned.quantity
        : workout.totalEnergyBurned;
      musclePoints += (energy || 0) * 100;
      // Multiply kcal by 100 as a rough scaling to get it into a comparable
range of "points"
    } else {
      // If no energy, use duration as proxy (e.g., 30 min strength = some
points)
      musclePoints += (workout.duration || 0) *
10; // 10 points per minute as a rough guess
    }
  }

  // 9. Combine cardio and muscle points, apply logarithmic strain scaling
  const totalLoad = cardioPoints + musclePoints;
  // Use a logarithmic formula to convert totalLoad to a 0–21 strain score
  // We add 1 to avoid log(0), and use a scaling factor for calibration.
  const LOG_BASE = Math.E;         // using natural log
  const SCALE =
```typescript
  const LOG_BASE = Math.E;         // using natural log for scaling
  const SCALE = 3;                 // scaling factor to map load to 0-21
(tunable)
  // Compute raw strain score via logarithm
  const strainScoreRaw = Math.log(totalLoad + 1);  // log_e(totalLoad + 1)
~~~

```
  let strainScore = SCALE * strainScoreRaw;
  // (Optionally, a baseline offset could be added here if desired, e.g. to
account for daily non-exercise stress)

  // 10. Cap the strain score to the 0–21 range and round for clarity
  if (strainScore > 21) strainScore = 21;
  if (strainScore < 0) strainScore = 0;
  // Round to one decimal place (WHOOP strain is typically reported with one
decimal)
  strainScore = Math.round(strainScore * 10) / 10;

  return strainScore;
}
```

In this implementation, we first gather all heart rate samples for the day using `HealthKit.queryQuantitySamples` (with heart rate in `"count/min"` units for BPM). We calculate the time spent in each heart rate zone and compute a weighted sum (`cardioPoints`). We then query workouts from HealthKit and filter those that correspond to strength training, accumulating a `musclePoints` total (using either logged volume from metadata or approximating via energy burned/duration). Finally, we combine the cardio and muscle points into `totalLoad` and apply a logarithmic scaling (with a chosen factor) to convert this into the strain score between 0 and 21. The result is a **Day Strain** value that reflects both the cardiovascular exertion (time in elevated heart rate zones) and muscular work done, on a logarithmic scale as described in WHOOP's methodology [4].

Note: The scaling constants (zone weights, the logarithm factor, etc.) can be adjusted based on empirical data or specific user calibration to closely match the expected 0–21 distribution. The above choices are heuristic, aiming to mirror the concept that **higher intensity and longer duration exponentially increase strain** while ensuring the score stays within the 0–21 range [4].

---

[1] [2] [3] [9] [10] [11] [12] [14] How Does WHOOP Strain Work? | WHOOP
https://www.whoop.com/us/en/thelocker/how-does-whoop-strain-work-101/?srsltid=AfmBOorQMG4NML1jXOq8iWLbbGUNlbq8h9K3PF0fe9uJqv1xZ2iBf3Yt

[4] [7] [8] I asked Whoop Coach to explain exactly how strain is calculated and show me the math : r/whoop
https://www.reddit.com/r/whoop/comments/1fg528j/i_asked_whoop_coach_to_explain_exactly_how_strain/

[5] Learn How to Calculate Your Max Heart Rate (MHR)
https://www.verywellfit.com/maximum-heart-rate-1231221

[6] [13] [15] Reputable Health
https://www.reputable.health/blog/how-does-whoop-calculate-strain