Package: PhysioMSKNet (via r-universe)

May 16, 2026

Type Package

Title Musculoskeletal Network Analysis for Physiological Data

Version 0.1.0

Author Yusuke Matsui

Maintainer Yusuke Matsui <`matsui@nagoya-u.jp`>

Description Network science analysis of the human musculoskeletal system. Implements hypergraph representations of muscle-bone connections, physical simulation via damped harmonic oscillators, perturbation-based impact scoring, community detection, and clinical outcome prediction. Based on Murphy et al. (2018) <doi:10.1371/journal.pbio.2002811>.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.2)

Imports Matrix, stats, graphics, grDevices, utils

Suggests igraph, MASS, ggplot2, readxl, xml2, PhysioAnnotationHub, PhysioCrossModal, PhysioEMG, PhysioMoCap, PhysioOpenSim, SummarizedExperiment, testthat (>= 3.0.0)

Config/testthat/edition 3

Repository https://x-biosignal.r-universe.dev

Date/Publication 2026-03-16 11:31:48 UTC

RemoteUrl https://github.com/x-biosignal/PhysioMSKNet

RemoteRef HEAD

RemoteSha bfd3d9051143545d39295baba79ad29d079ef9a3

Contents

`degreeDistribution` *Degree Distribution*

Description

Computes the degree distribution for bones or muscles.

Usage

```
degreeDistribution(hg, type = c("muscle", "bone"))
```

Arguments

`hg`	An MSKHypergraph object.
`type`	Character, either "muscle" (hyperedge degree) or "bone" (vertex degree).

Value

A data.frame with columns: degree, count, probability

`emgCommunityCompare` *Compare EMG and MSK Communities*

Description

Detects functional communities from EMG coherence and compares them with structural communities from the MSK network.

Usage

```
emgCommunityCompare(pe, hg = NULL, gamma = 4.3, mapping = NULL)
```

Arguments

`pe`	A SummarizedExperiment-like object or numeric signal matrix.
`hg`	An MSKHypergraph object (NULL loads default).
`gamma`	Resolution parameter for MSK community detection (default: 4.3).
`mapping`	Optional data.frame from `emgToMSKMapping()`.

Value

A list with:

emg_communities Named integer vector of EMG community assignments
msk_communities Named integer vector of MSK community assignments
z_rand z-Rand score comparing the two partitions
mapping The channel-to-muscle mapping used

`emgMSKEnrichment` *EMG Activation Enrichment by MSK Community*

Description

Tests whether EMG activation levels differ across MSK community assignments using a Kruskal-Wallis test.

Usage

```
emgMSKEnrichment(pe, hg = NULL, gamma = 4.3, mapping = NULL)
```

Arguments

`pe`	A SummarizedExperiment-like object or numeric signal matrix.
`hg`	An MSKHypergraph object (NULL loads default).
`gamma`	Resolution parameter for MSK community detection (default: 4.3).
`mapping`	Optional data.frame from `emgToMSKMapping()`.

Value

A list with:

per_community Data frame with community, mean/median activation

overall_test Kruskal-Wallis test result

activation_values Named numeric vector of RMS activation

`emgStructuralCoherence` *Compare EMG Coherence with MSK Structural Connectivity*

Description

Computes EMG functional coherence and compares it with structural adjacency from the MSK muscle graph using a Mantel test.

Usage

```
emgStructuralCoherence(pe, hg = NULL, freq_band = c(20, 50), mapping = NULL)
```

Arguments

`pe`	A SummarizedExperiment-like object or a numeric signal matrix (time x channels).
`hg`	An MSKHypergraph object (NULL loads default).
`freq_band`	Numeric vector of length 2, frequency band in Hz for coherence (default: c(20, 50) for EMG beta/gamma).
`mapping`	Optional data.frame from `emgToMSKMapping()`.

Value

A list with:

coherence_matrix EMG functional coherence matrix

structural_matrix MSK structural adjacency (matched subset)

correlation Mantel correlation coefficient

p_value Permutation-based p-value

mapped_muscles Names of matched muscles

Examples

```
## Not run:
result <- emgStructuralCoherence(pe_emg, freq_band = c(20, 50))

## End(Not run)
```

`emgToMSKMapping` *Map EMG Channels to MSK Muscles*

Description

Matches EMG channel names from a PhysioExperiment/SummarizedExperiment object to muscles in an MSK hypergraph.

Usage

```
emgToMSKMapping(pe, hg = NULL, method = c("exact", "fuzzy"), threshold = 0.8)
```

Arguments

`pe`	A SummarizedExperiment-like object with channel names in `colData(pe)$name` or `colnames(assay(pe))`.
`hg`	An MSKHypergraph object (NULL loads default).
`method`	Character, matching method: "exact" or "fuzzy".
`threshold`	Numeric, fuzzy matching threshold (default: 0.8).

Value

A data.frame with columns: channel_idx, channel_name, muscle_idx, muscle_name, match_quality.

Examples

```
## Not run:
mapping <- emgToMSKMapping(pe_emg, method = "fuzzy")

## End(Not run)
```

`hyperedgeDegree` *Hyperedge (Muscle) Degree*

Description

Returns the degree of each hyperedge (muscle), i.e., the number of bones each muscle attaches to.

Usage

```
hyperedgeDegree(hg)
```

Arguments

`hg`	An MSKHypergraph object or incidence matrix.

Value

Named numeric vector of muscle degrees.

`imuCommunityDynamics` *IMU-based Community Dynamics*

Description

Analyzes movement synchrony from IMU sensors in the context of MSK network communities. Computes within-community vs between-community sensor synchrony and tests for significance.

Usage

```
imuCommunityDynamics(
  imu_data,
  hg = NULL,
  gamma = 4.3,
  mapping = NULL,
  window_sec = NULL
)
```

Arguments

`imu_data`	Named list of per-sensor data (see `imuNetworkKinematics`).
`hg`	An MSKHypergraph object (NULL loads default).
`gamma`	Numeric, resolution parameter for community detection (default 4.3).
`mapping`	Pre-computed mapping (optional).
`window_sec`	Numeric or NULL. If provided, computes time-resolved synchrony in sliding windows of this duration (seconds). Requires `sampling_rate` attribute on sensor data.

Value

A list with:

within_community_sync Mean synchrony within communities

between_community_sync Mean synchrony between communities

ratio Within/between ratio

p_value Wilcoxon test p-value

time_resolved Data frame if window_sec provided, NULL otherwise

---

`imuImpactPrediction` *IMU-based Impact Prediction*

Description

Computes bone stress from IMU acceleration/angular velocity data and propagates it through the MSK incidence matrix to predict muscle vulnerability.

Usage

```
imuImpactPrediction(
  imu_data,
  hg = NULL,
  mapping = NULL,
  stress_metric = c("acceleration", "angular_velocity", "jerk", "composite"),
  use_proxy = TRUE
)
```

Arguments

`imu_data`	A named list of per-sensor data, or a matrix (see `imuNetworkKinematics` for formats).
`hg`	An MSKHypergraph object (NULL loads default).
`mapping`	Pre-computed mapping (optional).

`stress_metric`	Character: `"acceleration"` (peak linear acceleration), `"angular_velocity"` (peak angular velocity), `"jerk"` (peak rate of change of acceleration), `"composite"` (weighted combination of acceleration + angular velocity).
`use_proxy`	Logical, use degree-based proxy for impact deviation (faster) or run full simulation.

Value

A list with:

vulnerability Named numeric vector of muscle vulnerability scores

bone_stress Named numeric vector of bone stress

muscle_stress_exposure Named numeric of stress propagated to muscles

ranking Data frame ranked by vulnerability (descending)

`imuNetworkKinematics` *IMU-based Network Kinematics Analysis*

Description

Computes pairwise kinematic coupling from IMU orientation or acceleration data and compares it to the structural adjacency of the MSK bone graph using a Mantel test.

Usage

```
imuNetworkKinematics(
  imu_data,
  hg = NULL,
  mapping = NULL,
  signal = c("orientation", "acceleration"),
  method = c("correlation", "mutual_info")
)
```

Arguments

`imu_data`	A named list of per-sensor data. Each element should be a matrix or data.frame with orientation/acceleration columns. The list names are sensor placement names (e.g., `"upper_arm"`, `"thigh"`). Alternatively, a single matrix where columns are named `"<sensor>_roll"`, `"<sensor>_pitch"`, `"<sensor>_yaw"` or `"<sensor>_ax"`, `"<sensor>_ay"`, `"<sensor>_az"`.
`hg`	An MSKHypergraph object (NULL loads default).
`mapping`	A pre-computed mapping from `imuToMSKMapping()` (optional).
`signal`	Character, which signal to use for coupling: `"orientation"` (Euler angle differences) or `"acceleration"` (acceleration magnitude correlation).
`method`	Character, coupling method: `"correlation"` or `"mutual_info"`.

Value

A list with:

kinematic_coupling Pairwise coupling matrix between mapped bones

structural_matrix Corresponding MSK bone graph adjacency

correlation Mantel test correlation coefficient

p_value Permutation p-value

mapped_sensors Data frame of sensor-to-bone mapping used

`imuToMSKMapping` *Map IMU Sensor Placements to MSK Bones*

Description

Matches IMU sensor placement names to bones in the MSK hypergraph using a curated lookup table with fuzzy matching fallback. Supports naming conventions from Xsens, APDM, Shimmer, and generic body-segment labels.

Usage

```
imuToMSKMapping(
  sensors,
  hg = NULL,
  method = c("exact", "fuzzy"),
  threshold = 0.7
)
```

Arguments

`sensors`	Character vector of IMU sensor placement names (e.g., `c("upper_arm", "thigh", "lumbar")`).
`hg`	An MSKHypergraph object (NULL loads default).
`method`	Character, matching method: "exact" or "fuzzy".
`threshold`	Numeric, fuzzy matching threshold (default: 0.7).

Value

A data.frame with columns: sensor_name, bone_idx, bone_name, match_quality, match_method.

Examples

```
## Not run:
mapping <- imuToMSKMapping(c("upper_arm", "thigh", "lumbar"))

## End(Not run)
```

`loadIncidenceMatrix` *Load Incidence Matrix*

Description

Loads the bipartite incidence matrix C where rows are bones (vertices) and columns are muscles (hyperedges). Entry C[i,j] = 1 indicates that muscle j attaches to bone i.

Usage

```
loadIncidenceMatrix()
```

Value

A sparse Matrix (dgCMatrix) of dimensions 173 x 270

References

Murphy AC et al. (2018) PLOS Biology.

`loadMSKData` *Load MSK Network Data*

Description

Loads the complete musculoskeletal network dataset from Murphy et al. (2018). Returns a list containing the incidence matrix, muscle metadata, and validation data for reproducing the paper's results.

Usage

```
loadMSKData()
```

Value

A list with components:

incidence Sparse incidence matrix C (bones x muscles)

muscle_meta Data frame with muscle names, community assignments, homunculus categories

bone_names Character vector of bone/vertex names

muscle_names Character vector of muscle/hyperedge names

References

Murphy AC et al. (2018) "Structure, function, and control of the human musculoskeletal network." PLOS Biology 16(1): e2002811.

`loadMuscleMetadata` *Load Muscle Metadata*

Description

Loads metadata for all 270 muscles including community assignments and homunculus category labels from the paper.

Usage

```
loadMuscleMetadata()
```

Value

A data.frame with columns: index, muscle, community, homunculus_category

References

Murphy AC et al. (2018) PLOS Biology.

`loadValidationData` *Load Validation Data*

Description

Loads validation datasets used to reproduce key figures from the paper.

Usage

```
loadValidationData(
  dataset = c("degree_distribution", "impact_vs_recovery", "homunculus_deviation",
    "fmri_activation", "homunculus_coordinates", "impact_vs_path", "impact_scores")
)
```

Arguments

`dataset`	Character string specifying which dataset to load:
	"degree_distribution" Degree probability for real vs null hypergraphs (fig2e)
	"impact_vs_recovery" Recovery time vs impact deviation (fig3b)
	"homunculus_deviation" Homunculus area vs deviation ratio (fig4b)
	"fmri_activation" Impact deviation vs fMRI activation volume (fig4c)
	"homunculus_coordinates" Homunculus area vs muscle MDS coordinate (fig4d)
	"impact_vs_path" Average shortest path vs impact score (figS11)

Value

A data.frame with the requested validation data

References

Murphy AC et al. (2018) PLOS Biology.

`mocapCommunityDynamics` *Within vs Between Community Movement Synchrony*

Description

Computes displacement synchrony (correlation) between matched bones and tests whether within-community pairs are more synchronized than between-community pairs using a Wilcoxon test.

Usage

```
mocapCommunityDynamics(
  pe_mocap,
  hg = NULL,
  gamma = 4.3,
  mapping = NULL,
  window_sec = NULL
)
```

Arguments

`pe_mocap`	A numeric matrix (frames x segments) with MoCap data.
`hg`	An MSKHypergraph object (NULL loads default).
`gamma`	Resolution parameter for community detection (default: 4.3).
`mapping`	Optional data.frame from `mocapToMSKMapping()`.
`window_sec`	Optional numeric, window size in seconds for time-resolved analysis (NULL for global analysis only).

Value

A list with:

within_community_sync Mean within-community synchrony

between_community_sync Mean between-community synchrony

ratio Within/between ratio

p_value Wilcoxon test p-value

time_resolved Data frame with per-window results (if window_sec set)

`mocapImpactPrediction` *Kinematic Stress-based Impact Prediction*

Description

Computes kinematic stress per bone from MoCap data and propagates it through the MSK incidence matrix to estimate muscle vulnerability.

Usage

```
mocapImpactPrediction(
  pe_mocap,
  hg = NULL,
  mapping = NULL,
  stress_metric = c("acceleration", "jerk", "range"),
  use_proxy = TRUE
)
```

Arguments

`pe_mocap`	A numeric matrix (frames x segments) with MoCap data.
`hg`	An MSKHypergraph object (NULL loads default).
`mapping`	Optional data.frame from `mocapToMSKMapping()`.
`stress_metric`	Character, kinematic stress metric: "acceleration", "jerk", or "range".
`use_proxy`	Logical, if TRUE uses degree-based proxy instead of full simulation for impact deviation (default: TRUE).

Value

A list with:

vulnerability Named numeric vector of muscle vulnerability scores

bone_stress Named numeric vector of bone stress values

muscle_stress_exposure Named numeric vector of muscle stress exposure

ranking Data frame ranking muscles by vulnerability

`mocapNetworkKinematics` *Compute Kinematic Coupling vs MSK Structure*

Description

Computes pairwise movement coupling between MoCap segments and compares the kinematic coupling matrix with MSK structural adjacency.

Usage

```
mocapNetworkKinematics(
  pe_mocap,
  hg = NULL,
  mapping = NULL,
  method = c("correlation", "mutual_info")
)
```

Arguments

`pe_mocap`	A numeric matrix (frames x segments) or SummarizedExperiment with MoCap position data.
`hg`	An MSKHypergraph object (NULL loads default).
`mapping`	Optional data.frame from `mocapToMSKMapping()`.
`method`	Character, coupling method: "correlation" or "mutual_info".

Value

A list with:

kinematic_coupling Pairwise coupling matrix

structural_matrix MSK bone adjacency (matched subset)

correlation Mantel correlation coefficient

p_value Permutation-based p-value

`mocapToMSKMapping` *Map MoCap Segments to MSK Bones*

Description

Matches MoCap segment names from a SkeletonModel or character vector to bones in an MSK hypergraph using a curated lookup table with fuzzy matching fallback.

Usage

```
mocapToMSKMapping(
  skeleton,
  hg = NULL,
  method = c("exact", "fuzzy"),
  threshold = 0.7
)
```

Arguments

`skeleton`	A PhysioMoCap SkeletonModel object or character vector of segment names.
`hg`	An MSKHypergraph object (NULL loads default).
`method`	Character, matching method: "exact" or "fuzzy".
`threshold`	Numeric, fuzzy matching threshold (default: 0.7).

Value

A data.frame with columns: segment_name, bone_idx, bone_name, match_quality, match_method.

Examples

```
## Not run:
mapping <- mocapToMSKMapping(c("upper_arm", "forearm", "thigh"))

## End(Not run)
```

`mskAdaptProtocol` *Adapt Rehabilitation Protocol Based on Reassessment*

Description

Modifies a rehabilitation protocol based on patient reassessment results. Uses decision rules to determine whether to advance, continue, reduce, or modify the current protocol.

Usage

```
mskAdaptProtocol(reassessment, current_protocol, hg = NULL)
```

Arguments

`reassessment`	An `MSKReassessment` object.
`current_protocol`	An `MSKRehabProtocol` object from `mskRehabProtocol()`.
`hg`	An MSKHypergraph object (NULL loads default).

Value

An S3 object of class `"MSKAdaptedProtocol"` with:

decision character: "advance", "continue", "reduce", "modify"

adapted_exercises data.frame with modified exercise prescription

rationale character explanation

focus_muscles character vector of muscles needing attention

removed_exercises exercises to drop

added_exercises new exercises to add

Examples

```
## Not run:
protocol <- mskRehabProtocol("Biceps Brachii")
adapted <- mskAdaptProtocol(reassessment, protocol)
print(adapted)

## End(Not run)
```

`mskAnnotate` *Annotate MSK Hypergraph with Knowledge Graph Data*

Description

Attaches anatomical annotations from PhysioAnnotationHub to an MSK hypergraph, matching muscle and bone names between the two data sources. Annotations include body region, innervation, actions, and spinal levels.

Usage

```
mskAnnotate(hg = NULL, hub = NULL)
```

Arguments

`hg`	An MSKHypergraph object. If NULL, loads default 173-bone/270-muscle network.
`hub`	A PhysioAnnotationHub object. If NULL, loads via `PhysioAnnotationHub::loadAnnotationHub()`.

Value

An annotated MSKHypergraph with additional fields:

muscle_annotations Data frame of muscle annotations merged from hub

bone_annotations Data frame of bone annotations merged from hub

annotated Logical flag indicating annotations are attached

annotation_coverage List with muscle and bone match rates

Clinical Validity

Annotations are derived from PhysioAnnotationHub's curated knowledge graph. Name matching uses fuzzy matching with a 0.2 edit distance threshold, which may produce incorrect matches for similarly named structures. Always verify critical annotations against primary anatomical references.

References

Murphy AC et al. (2018) PLOS Biology 16(1): e2002811.

Examples

```
## Not run:
hg <- mskAnnotate()
head(hg$muscle_annotations)

## End(Not run)
```

`mskBetweenness` *Betweenness Centrality via BFS*

Description

Computes betweenness centrality for the projected graph. Uses igraph if available, otherwise a pure-R BFS implementation.

Usage

```
mskBetweenness(hg, type = c("bone", "muscle"))
```

Arguments

`hg`	An MSKHypergraph object.
`type`	Character, "bone" or "muscle" projection.

Value

Named numeric vector of betweenness centrality values.

`mskClinicalEvidence` *Map Injured Muscles to Clinical Codes and Evidence*

Description

Generates a clinical evidence report for a set of injured muscles by querying the knowledge graph for ICD-10 codes, ICF codes, innervation, functional impact, and related muscles (synergists and antagonists).

Usage

```
mskClinicalEvidence(injury_muscles, hub = NULL, hg = NULL)
```

Arguments

`injury_muscles`	Character or integer vector identifying injured muscles.
`hub`	A PhysioAnnotationHub object (NULL loads default).
`hg`	An MSKHypergraph object (NULL loads default).

Value

An S3 object of class `"MSKClinicalEvidence"` with:

muscles Data frame of injury muscle annotations

icd10_codes Matching ICD-10 entries

icf_codes Matching ICF entries

affected_nerves Nerves innervating injured muscles

affected_spinal_levels Spinal segments involved

functional_impact Affected actions/movements

synergists Synergistic muscles from KG

antagonists Antagonistic muscles from KG

Clinical Validity

ICD-10 and ICF code mappings are based on knowledge graph associations and may not capture all valid codes for a clinical scenario. This tool provides evidence aggregation for research; clinical coding should follow local guidelines and be reviewed by qualified professionals.

Examples

```
## Not run:
evidence <- mskClinicalEvidence(c("Biceps Brachii", "Deltoid"))
print(evidence)

## End(Not run)
```

`mskClinicalPredictor` *Clinical Prediction from MSK Network Topology*

Description

Predicts recovery time, identifies compensatory muscles, and estimates secondary injury risk based on MSK network impact analysis.

Usage

```
mskClinicalPredictor(injury_muscles, hg = NULL, sim = NULL, verbose = TRUE)
```

Arguments

`injury_muscles`	Character or integer vector identifying injured muscles.
`hg`	An MSKHypergraph object (NULL loads default 173-bone/270-muscle network).
`sim`	An MSKSimulation object (NULL creates one with default parameters).
`verbose`	Logical, print progress messages (default: TRUE).

Value

An S3 object of class `"MSKClinicalPrediction"` with:

recovery Data frame with predicted recovery weeks and CI per muscle

compensatory List of compensatory muscles per injured muscle

secondary_risk Data frame of secondary injury risk scores

injury_muscles Resolved muscle names

injury_indices Resolved integer indices

Clinical Validity

The recovery model was validated on 14 aggregate muscle groups, not individual muscles. Patient factor adjustments are heuristic, not independently validated. This is a research exploration tool, not a clinical diagnostic.

References

Murphy AC et al. (2018) PLOS Biology 16(1): e2002811.

Examples

```
## Not run:
pred <- mskClinicalPredictor(c("Biceps Brachii", "Deltoid"))
print(pred)

## End(Not run)
```

`mskCloseness` *Closeness Centrality*

Description

Closeness Centrality

Usage

```
mskCloseness(hg, type = c("bone", "muscle"))
```

Arguments

`hg`	An MSKHypergraph object.
`type`	Character, "bone" or "muscle" projection.

Value

Named numeric vector of closeness centrality values.

`mskCommunityDetect` *Community Detection on MSK Network*

Description

Detects communities in the projected muscle-muscle graph using the Louvain algorithm with a resolution parameter gamma.

Usage

```
mskCommunityDetect(hg = NULL, gamma = 4.3, type = c("muscle", "bone"))
```

Arguments

`hg`	An MSKHypergraph object. If NULL, loads built-in data.
`gamma`	Resolution parameter for modularity (default: 4.3, as in paper). Higher values produce more, smaller communities.
`type`	Projection type: "muscle" (default, as in paper) or "bone".

Value

A list with:

membership Named integer vector of community assignments

n_communities Number of communities detected

modularity Modularity value Q

gamma Resolution parameter used

sizes Table of community sizes

References

Murphy AC et al. (2018) PLOS Biology 16(1): e2002811.

`mskCommunityProfile` *Profile a MSK Community's Functional Characteristics*

Description

Generates a detailed functional profile of a musculoskeletal community by combining network topology metrics with knowledge graph annotations. This enables characterization of communities by their anatomical region, innervation patterns, primary actions, and spinal segment involvement.

Usage

```
mskCommunityProfile(hg = NULL, community_id, hub = NULL, gamma = 4.3)
```

Arguments

`hg`	An MSKHypergraph object. If NULL, loads default network.
`community_id`	Integer, the community to profile (1-based).
`hub`	A PhysioAnnotationHub object (NULL loads default).
`gamma`	Numeric, resolution parameter for community detection (default: 4.3, as in Murphy et al. 2018).

Value

An S3 object of class `"MSKCommunityProfile"` with:

community_id Integer, the profiled community

muscles Character vector of muscle names in the community

n_muscles Integer, number of muscles

action_profile Table of primary action distribution

nerve_profile Table of innervating nerves distribution

region_profile Table of body region distribution

spinal_profile Table of spinal level distribution

dominant_action Most frequent primary action

dominant_nerve Most frequent innervating nerve

dominant_region Most frequent body region

mean_degree Mean hyperedge degree of community muscles

mean_impact_deviation Mean impact deviation score

Clinical Validity

Community membership depends on the resolution parameter gamma and the stochastic Louvain algorithm. Profile annotations depend on KG completeness. Community boundaries are network-derived, not anatomical boundaries.

References

Murphy AC et al. (2018) PLOS Biology 16(1): e2002811.

Examples

```
## Not run:
profile <- mskCommunityProfile(community_id = 1)
print(profile)

## End(Not run)
```

`mskCompensationEvolution`

Track Compensation Pattern Evolution Over Time

Description

Tracks changes in compensation patterns across multiple timepoints, identifying onset, resolution, and trends.

Usage

```
mskCompensationEvolution(
  timepoints_emg,
  injured_muscles,
  hg = NULL,
  emg_mapping = NULL,
  sr = NULL,
  z_threshold = 1.96
)
```

Arguments

`timepoints_emg`	Named list of EMG matrices. The first element is treated as baseline.
`injured_muscles`	Character vector of injured muscle names or integer indices.
`hg`	An MSKHypergraph object (NULL loads default).
`emg_mapping`	Optional pre-computed data.frame from `emgToMSKMapping()`.
`sr`	Optional sampling rate override.
`z_threshold`	Numeric, z-score threshold (default: 1.96).

Value

An S3 object of class `"MSKCompensationEvolution"` with:

evolution_table Data.frame of per-timepoint per-muscle z-scores

onset_timepoint Per-muscle first timepoint of compensation

resolution_timepoint Per-muscle first timepoint of resolution

trend Per-muscle trend classification

summary Data.frame of per-timepoint summary statistics

Examples

```
## Not run:
evolution <- mskCompensationEvolution(
  timepoints_emg = list(week0 = emg0, week2 = emg2, week4 = emg4),
  injured_muscles = c("Biceps Brachii")
)

## End(Not run)
```

`mskCompensationNetwork` *Build Compensation Network*

Description

Constructs a compensation-weighted subgraph from compensation detection results, identifying compensation chains and hub muscles.

Usage

```
mskCompensationNetwork(compensation_result, hg = NULL, emg_mapping = NULL)
```

Arguments

`compensation_result`	An `"MSKCompensation"` object.
`hg`	An MSKHypergraph object (NULL loads default).
`emg_mapping`	Optional pre-computed data.frame from `emgToMSKMapping()`.

Value

A list with:

adjacency Weighted compensation adjacency matrix

chains List of compensation chains

hub_muscles Muscles in multiple compensation chains

community_involvement MSK communities affected by compensation

Examples

```
## Not run:
net <- mskCompensationNetwork(compensation_result, hg)

## End(Not run)
```

`mskCompensationRiskScore`

Compensation Risk Score

Description

Computes risk scores for compensating muscles based on biomechanical overuse principles, integrating network topology, duration of compensation, and loading intensity.

Usage

```
mskCompensationRiskScore(
  compensation_result,
  hg = NULL,
  duration_weeks = 0,
  load_intensity = c("low", "moderate", "high")
)
```

Arguments

`compensation_result`	An `"MSKCompensation"` object from `mskDetectCompensation()`.
`hg`	An MSKHypergraph object (NULL loads default).
`duration_weeks`	Numeric, how long compensation has been occurring.
`load_intensity`	Character, one of "low", "moderate", "high".

Value

An S3 object of class `"MSKCompensationRisk"` with:

per_muscle_risk Data.frame of per-muscle risk scores

overall_risk Numeric overall risk score

overall_category Character risk category

highest_risk_muscle Name of highest-risk muscle

recommendation Clinical action recommendation

Risk Model

Per compensating muscle: risk = z_excess * degree_normalized * duration_factor * load_factor where z_excess = max(0, z_score - z_threshold), degree_normalized = hyperedgeDegree / mean_degree, duration_factor = 1 + log(1 + duration_weeks), load_factor = intensity multiplier (0.5, 1.0, 1.5).

Examples

```
## Not run:
risk <- mskCompensationRiskScore(compensation_result,
  duration_weeks = 4, load_intensity = "moderate")

## End(Not run)
```

`mskCompensationSummary`

Comprehensive Compensation Analysis Summary

Description

Orchestrator that runs all compensation analyses and returns a unified summary. Uses tryCatch for each sub-analysis so partial results are available even if some analyses fail.

Usage

```
mskCompensationSummary(
  emg,
  emg_baseline,
  injured_muscles,
  timepoints_emg = NULL,
  hg = NULL,
  sr = NULL,
  z_threshold = 1.96,
  duration_weeks = 0,
  load_intensity = c("low", "moderate", "high")
)
```

Arguments

`emg`	Current EMG data (matrix or SummarizedExperiment).
`emg_baseline`	Baseline/pre-injury EMG data.
`injured_muscles`	Character vector of injured muscle names or integer indices.
`timepoints_emg`	Optional named list of EMG matrices for evolution analysis.
`hg`	An MSKHypergraph object (NULL loads default).
`sr`	Optional sampling rate override.
`z_threshold`	Numeric, z-score threshold (default: 1.96).
`duration_weeks`	Numeric, compensation duration for risk scoring.
`load_intensity`	Character, load intensity for risk scoring.

Value

An S3 object of class `"MSKCompensationSummary"` with sub-results and available_analyses vector.

Examples

```
## Not run:
summary <- mskCompensationSummary(
  emg = emg_current, emg_baseline = emg_pre,
  injured_muscles = c("Biceps Brachii"),
  duration_weeks = 4, load_intensity = "moderate"
)

## End(Not run)
```

`mskConsensusPartition` *Consensus Partition*

Description

Runs Louvain community detection multiple times and extracts a consensus partition. The paper uses 100 runs and takes the most frequent partition for each node pair.

Usage

```
mskConsensusPartition(
  hg = NULL,
  gamma = 4.3,
  n_runs = 100L,
  type = c("muscle", "bone")
)
```

Arguments

`hg`	An MSKHypergraph object.
`gamma`	Resolution parameter (default: 4.3).
`n_runs`	Number of runs for consensus (default: 100).
`type`	Projection type: "muscle" or "bone".

Value

Same structure as mskCommunityDetect, but with consensus partition.

References

Murphy AC et al. (2018) PLOS Biology.

`mskCoordinationQualityScore` *Compute Coordination Quality Score*

Description

Compares a subject's EMG coordination pattern against a reference (healthy baseline or normative data) to produce a 0-1 quality score.

Usage

```
mskCoordinationQualityScore(
  emg,
  reference_emg = NULL,
  hg = NULL,
  method = c("synergy_distance", "correlation_profile", "network_similarity")
)
```

Arguments

`emg`	EMG data: matrix (time x channels), SummarizedExperiment, or vector.
`reference_emg`	Reference EMG matrix (or NULL for within-subject reference).
`hg`	An MSKHypergraph object (NULL loads default).
`method`	Character, comparison method: "synergy_distance" (default), "correlation_profile", or "network_similarity".

Value

A list with: quality_score (0-1, 1=perfect match to reference), component_scores, muscle_contributions.

Examples

```
## Not run:
cqs <- mskCoordinationQualityScore(emg, reference_emg, method = "synergy_distance")

## End(Not run)
```

`mskDetectCompensation` *Detect Compensatory Activation Patterns*

Description

Detects compensatory movement patterns by comparing current EMG activation against a baseline, using MSK network topology to identify muscles that are structurally positioned to compensate for injured muscles.

Usage

```
mskDetectCompensation(
  emg,
  emg_baseline,
  injured_muscles,
  hg = NULL,
  emg_mapping = NULL,
  sr = NULL,
  z_threshold = 1.96,
  neighborhood_order = 2L
)
```

Arguments

`emg`	Current EMG data (matrix, SummarizedExperiment, or numeric vector).
`emg_baseline`	Baseline/pre-injury EMG data (same format as `emg`).
`injured_muscles`	Character vector of injured muscle names or integer indices.
`hg`	An MSKHypergraph object (NULL loads default).
`emg_mapping`	Optional pre-computed data.frame from `emgToMSKMapping()`.
`sr`	Optional sampling rate override.
`z_threshold`	Numeric, z-score threshold for flagging compensation (default: 1.96).
`neighborhood_order`	Integer, MSK graph distance to search for compensators (default: 2).

Value

An S3 object of class `"MSKCompensation"` with:

compensating_muscles Data.frame of muscles showing compensatory activation

injured_status Data.frame of injured muscle activation status

non_compensating Data.frame of neighbors that did NOT compensate

compensation_prevalence Proportion of neighbors showing compensation

network_context List with neighborhood info

Algorithm

1. Map EMG channels to MSK muscles via emgToMSKMapping
2. Compute RMS activation for both current and baseline
3. Compute z-score of change: z = (current_rms - baseline_rms) / baseline_sd
4. Identify MSK neighbors of injured muscles within `neighborhood_order` hops
5. Flag muscles where they are neighbors AND z-score > z_threshold
6. Also detect decreased activation in injured muscles (z < -z_threshold)

Examples

```
## Not run:
result <- mskDetectCompensation(
  emg = emg_current, emg_baseline = emg_pre,
  injured_muscles = c("Biceps Brachii"),
  z_threshold = 1.96
)
print(result)

## End(Not run)
```

---

mskDetectRecoveryPlateau

Detect Recovery Plateau in Longitudinal Data

Description

Identifies when recovery has plateaued using rolling window slope analysis or change rate assessment.

Usage

```
mskDetectRecoveryPlateau(
  tracker,
  window = 3L,
  min_slope = NULL,
  method = c("slope", "change_rate")
)
```

Arguments

`tracker`	An `MSKLongitudinalTracker` object.
`window`	Integer, number of consecutive timepoints to assess (default: 3).
`min_slope`	Numeric, minimum slope to be considered "improving" (default: NULL, auto-computed from data variability).
`method`	Character, "slope" (default) or "change_rate".

Value

An S3 object of class `"MSKRecoveryPlateau"` with:

plateau_detected logical per muscle

plateau_onset timepoint index where plateau begins (NA if none)

details data.frame with per-window slopes

recommendation character ("continue", "modify_protocol", "reassess")

Examples

```
## Not run:
plateau <- mskDetectRecoveryPlateau(tracker, window = 3)

## End(Not run)
```

`mskDetectResponderStatus`

Classify Muscles as Responders, Non-Responders, or Deteriorated

Description

Compares first and last timepoint to determine if each muscle shows clinically meaningful change beyond the MDC threshold.

Usage

```
mskDetectResponderStatus(
  tracker,
  mdc_result = NULL,
  threshold_type = c("mdc", "effect_size")
)
```

Arguments

`tracker`	An `MSKLongitudinalTracker` object.
`mdc_result`	An `MSKMinimalDetectableChange` object (or NULL to use effect size threshold).
`threshold_type`	Character, "mdc" (default) or "effect_size" (Cohen's d > 0.8).

Value

An S3 object of class `"MSKResponderStatus"` with:

classification data.frame (muscle, status, change, threshold, effect_size_d)

summary counts of responders/non-responders/deteriorated

overall_status "responder" if majority of muscles improve

Examples

```
## Not run:
status <- mskDetectResponderStatus(tracker, mdc_result)

## End(Not run)
```

`mskEnrichKG` *Functional Enrichment of Muscles via Knowledge Graph*

Description

Tests whether a set of muscles is enriched for specific anatomical or functional annotations (actions, nerves, body regions, spinal levels) compared to the full background set of muscles in the hypergraph.

Usage

```
mskEnrichKG(
  muscles,
  annotation_type = c("action", "nerve", "body_region", "spinal_level"),
  hub = NULL,
  hg = NULL
)
```

Arguments

`muscles`	Character or integer vector identifying muscles to test.
`annotation_type`	Character, one of `"action"`, `"nerve"`, `"body_region"`, or `"spinal_level"`. Partial matching is supported.
`hub`	A PhysioAnnotationHub object (NULL loads default).
`hg`	An MSKHypergraph object (NULL loads default).

Value

A data.frame with columns:

term Annotation term

count Number of query muscles with this term

background_count Number of background muscles with this term

expected Expected count under null

fold_enrichment Ratio of observed to expected

p_value P-value from hypergeometric test

significant Logical, TRUE if $p < 0.05$

Clinical Validity

Enrichment analysis depends on the completeness and accuracy of the underlying knowledge graph annotations. P-values are not adjusted for multiple testing; consider applying Bonferroni or FDR correction when testing multiple annotation types simultaneously.

Examples

```
## Not run:
# Test enrichment of upper limb muscles for nerve innervation
enrichment <- mskEnrichKG(
  c("Biceps Brachii", "Deltoid", "Trapezius"),
  annotation_type = "nerve"
)
enrichment[enrichment$significant, ]

## End(Not run)
```

`mskFunctionalMilestones` *Generate Functional Milestones for Rehabilitation*

Description

Takes the output of `mskPredictFunctionalOutcome` and generates time-based milestones for tracking rehabilitation progress.

Usage

```
mskFunctionalMilestones(
  outcome_prediction,
  n_milestones = 4L,
  milestone_type = c("clinical", "linear", "accelerating")
)
```

Arguments

`outcome_prediction`	An `MSKFunctionalOutcome` object.
`n_milestones`	Integer, number of milestones to generate (default: 4).
`milestone_type`	Character: "linear" (evenly spaced), "accelerating" (front-loaded), or "clinical" (based on standard rehab phases).

Value

An S3 object of class `"MSKFunctionalMilestones"` with:

milestones data.frame with milestone_id, week, target_rom, target_strength, target_function, phase_name, description

timeline_weeks total timeline in weeks

milestone_type character

Examples

```
## Not run:
outcome <- mskPredictFunctionalOutcome("Biceps Brachii")
milestones <- mskFunctionalMilestones(outcome, milestone_type = "clinical")
print(milestones)

## End(Not run)
```

`mskHomuncCorrelation` *Homunculus Correlation Analysis*

Description

Tests the correspondence between MSK network community structure and the motor cortex homunculus. Reproduces Fig 4b results. Target: $F(1,19) = 21.3$, $R^2 = 0.52$, $p < 0.001$

Usage

```
mskHomuncCorrelation(membership = NULL, homunculus_data = NULL)
```

Arguments

`membership`	Named integer vector of community assignments.
`homunculus_data`	Optional data.frame with columns: homunc_area, dev_ratio. If NULL, loads the built-in validation data.

Value

A list with regression result and deviation ratio by category.

References

Murphy AC et al. (2018) PLOS Biology.

`MSKHypergraph` *Create MSK Hypergraph Object*

Description

Constructs a musculoskeletal hypergraph from an incidence matrix. In this hypergraph, bones are vertices and muscles are hyperedges. A muscle (hyperedge) connects all bones it attaches to.

Usage

```
MSKHypergraph(C = NULL, muscle_meta = NULL)
```

Arguments

`C`	A matrix or sparse Matrix (bones x muscles) where C[i,j]=1 means muscle j attaches to bone i. If NULL, loads the built-in data.
`muscle_meta`	Optional data.frame with muscle metadata.

Value

An S3 object of class "MSKHypergraph" containing:

C Sparse incidence matrix (bones x muscles)

n_bones Number of bones (vertices)

n_muscles Number of muscles (hyperedges)

bone_names Character vector of bone names

muscle_names Character vector of muscle names

muscle_meta Muscle metadata if provided

References

Murphy AC et al. (2018) PLOS Biology 16(1): e2002811.

`mskImpactDeviation` *Compute Impact Deviation*

Description

Computes the impact deviation for each muscle, which is the difference between the observed impact score and the expected impact score for a muscle of that degree, expressed in standard deviations.

Usage

```
mskImpactDeviation(impact_scores, hg, null_scores = NULL)
```

Arguments

`impact_scores`	Named numeric vector of impact scores (from mskImpactScoreAll).
`hg`	An MSKHypergraph object.
`null_scores`	Optional matrix of null model impact scores (muscles x null_runs). If NULL, deviation is computed relative to a degree-based regression.

Details

The expected impact score is estimated from null model simulations or from a regression of impact score vs. degree.

Value

A named numeric vector of impact deviations.

References

Murphy AC et al. (2018) PLOS Biology.

`mskImpactRecoveryModel` *Impact-Recovery Prediction Model*

Description

Reproduces the key result from Fig 3b: correlation between impact deviation and clinical muscle injury recovery time. Target: $F(1,12) = 37.3$, $R^2 = 0.757$, $p < 0.0001$

Usage

```
mskImpactRecoveryModel(impact_deviation = NULL, recovery_data = NULL)
```

Arguments

`impact_deviation`	Named numeric vector of impact deviations.
`recovery_data`	Optional data.frame with columns: recovery_time, impact_deviation, weight. If NULL, loads the built-in validation data.

Value

Result from mskRobustRegression with additional paper comparison.

References

Murphy AC et al. (2018) PLOS Biology Table 4.

`mskImpactScore` *Compute Impact Score for One Muscle*

Description

Perturbs a single muscle and simulates the network dynamics to compute the total displacement of all bones (impact score). The perturbation is applied in the 4th spatial dimension to avoid directional artifacts.

Usage

```
mskImpactScore(sim, muscle_index)
```

Arguments

`sim`	An MSKSimulation object.
`muscle_index`	Integer index of the muscle to perturb.

Value

Numeric impact score (total displacement summed over all bones).

References

Murphy AC et al. (2018) PLOS Biology.

`mskImpactScoreAll` *Compute Impact Scores for All Muscles*

Description

Runs perturbation analysis for all 270 muscles and returns impact scores. This is the main computational function for reproducing Fig. 3a of the paper.

Usage

```
mskImpactScoreAll(sim = NULL, verbose = TRUE)
```

Arguments

`sim`	An MSKSimulation object. If NULL, creates one with default parameters.
`verbose`	Logical, print progress (default: TRUE).

Value

A named numeric vector of impact scores for each muscle.

References

Murphy AC et al. (2018) PLOS Biology.

`mskInjuryRiskProfile` *Patient-specific Injury Risk Profile*

Description

Computes a personalized injury risk profile for all muscles based on MSK network topology and patient characteristics.

Usage

```
mskInjuryRiskProfile(patient_data, hg = NULL, sim = NULL)
```

Arguments

`patient_data`	A list with patient characteristics: **age** Numeric, patient age in years (required) **bmi** Numeric, body mass index (optional) **activity_level** Character: "sedentary", "moderate", "active", "elite" (optional) **prior_injuries** Character vector of previously injured muscle names (optional)
`hg`	An MSKHypergraph object (NULL loads default).
`sim`	An MSKSimulation object (NULL creates default).

Value

An S3 object of class `"MSKInjuryRiskProfile"` with:

risk_scores Data frame with muscle name, base risk, adjusted risk

patient_data Input patient data

factors Applied adjustment factors

Clinical Validity

Patient factor adjustments (age, BMI, activity level) are heuristic multipliers, not derived from validated epidemiological models. This is a research exploration tool, not a clinical diagnostic.

Examples

```
## Not run:
profile <- mskInjuryRiskProfile(list(age = 45, activity_level = "active"))

## End(Not run)
```

`mskKGSummary` *KG-Enriched Summary of MSK Network*

Description

Produces a comprehensive summary of the MSK network annotated with knowledge graph data. Profiles all communities, identifies cross-community nerve pathways, and summarizes annotation coverage.

Usage

```
mskKGSummary(hg = NULL, hub = NULL, gamma = 4.3)
```

Arguments

`hg`	An MSKHypergraph object (NULL loads default).
`hub`	A PhysioAnnotationHub object (NULL loads default).
`gamma`	Numeric, resolution parameter for community detection (default: 4.3).

Value

An S3 object of class `"MSKKGSummary"` with:

hg Annotated MSKHypergraph

community_profiles List of MSKCommunityProfile objects

n_communities Number of detected communities

cross_community_nerves Data frame of nerves spanning communities

annotation_coverage Annotation match statistics

Clinical Validity

This is a summary tool combining network topology with knowledge graph annotations. Community boundaries and annotation mappings are model-derived. Use as a research exploration tool, not for clinical decision-making.

Examples

```
## Not run:
summary <- mskKGSummary()
print(summary)

## End(Not run)
```

`mskLongitudinalTracker`

Track Longitudinal MSK Metrics Across Timepoints

Description

Computes per-timepoint RMS activation, synergy decomposition (W/H/VAF), and optionally CMC for a series of EMG measurements taken at different timepoints during rehabilitation.

Usage

```
mskLongitudinalTracker(
  timepoints,
  hg = NULL,
  emg_mapping = NULL,
  metrics = c("rms", "synergy"),
  sr = NULL
)
```

Arguments

`timepoints`	Named list of EMG matrices (time x channels). Names should be timepoint labels (e.g., "T0", "T1", "T2").
`hg`	An MSKHypergraph object (NULL loads default).
`emg_mapping`	Optional pre-computed data.frame from `emgToMSKMapping()`.
`metrics`	Character vector of metrics to compute (default: c("rms", "synergy")). Options: "rms", "synergy", "mean_activation", "peak_activation".
`sr`	Optional sampling rate in Hz (overrides detected values).

Value

An S3 object of class "`MSKLongitudinalTracker`" with:

metrics_table data.frame (timepoint, muscle, metric_name, value)

timepoint_labels character vector

n_timepoints integer

synergy_series list of synergy results per timepoint

activation_series matrix (n_muscles x n_timepoints)

Examples

```
## Not run:
emg_t0 <- matrix(abs(rnorm(400)), 100, 4)
emg_t1 <- matrix(abs(rnorm(400)), 100, 4)
colnames(emg_t0) <- colnames(emg_t1) <- paste0("muscle_", 1:4)
tracker <- mskLongitudinalTracker(list(T0 = emg_t0, T1 = emg_t1))

## End(Not run)
```

`mskMinimalDetectableChange` *Compute Minimal Detectable Change from Longitudinal Tracker*

Description

Computes ICC (intraclass correlation coefficient) across timepoints for each metric, then derives SEM and MDC values for clinimetric assessment.

Usage

```
mskMinimalDetectableChange(
  tracker,
  method = "standard",
  confidence = 0.95,
  icc_method = "ICC(2,1)"
)
```

Arguments

`tracker`	An `MSKLongitudinalTracker` object.
`method`	Character, method for MDC computation (default: "standard").
`confidence`	Numeric, confidence level for MDC (default: 0.95).
`icc_method`	Character, ICC formula variant (default: "ICC(2,1)").

Value

An S3 object of class `"MSKMinimalDetectableChange"` with:

mdc_table data.frame (muscle, metric, ICC, SEM, MDC, pooled_SD)

confidence_level numeric

method character

Examples

```
## Not run:
mdc <- mskMinimalDetectableChange(tracker, confidence = 0.95)

## End(Not run)
```

`mskModularity` *Compute Modularity*

Description

Computes the modularity Q of a partition with resolution parameter. Q = (1/2m) * sum_ij (A_ij - gamma * k_i * k_j / (2m)) * delta(c_i, c_j)

Usage

```
mskModularity(A, membership, gamma = 1)
```

Arguments

`A`	Adjacency matrix.
`membership`	Integer vector of community assignments.
`gamma`	Resolution parameter (default: 1.0).

Value

Numeric modularity value.

`mskNetworkMetrics` *MSK Network Metrics*

Description

Computes standard graph metrics for the projected bone or muscle graph.

Usage

```
mskNetworkMetrics(hg, type = c("bone", "muscle"))
```

Arguments

`hg`	An MSKHypergraph object.
`type`	Character, "bone" or "muscle" projection.

Value

A list with: degree, strength, clustering_coef, density

`mskNeuralAdaptationIndex` *Compute Neural Adaptation Index Across Timepoints*

Description

Compares corticomuscular coherence (CMC) and directional coupling between two timepoints to quantify neural adaptation during rehabilitation.

Usage

```
mskNeuralAdaptationIndex(
  cmc_t0,
  cmc_t1,
  directional_t0 = NULL,
  directional_t1 = NULL
)
```

Arguments

`cmc_t0`	CMC result at baseline (MSKNeuromechCMC object or coherence matrix).
`cmc_t1`	CMC result at follow-up.
`directional_t0`	Directional coupling at baseline (MSKNeuromechDirectional object or NULL).
`directional_t1`	Directional coupling at follow-up (or NULL).

Value

A list with: cmc_change, directional_change, adaptation_index, interpretation ("improving"/"stable"/"declining").

Examples

```
## Not run:
nai <- mskNeuralAdaptationIndex(cmc_t0, cmc_t1, dir_t0, dir_t1)

## End(Not run)
```

`mskNullEnsemble` *Generate Null Hypergraph Ensemble*

Description

Creates multiple null hypergraphs and computes impact scores for each. This is used to compute impact deviations relative to the null distribution.

Usage

```
mskNullEnsemble(hg = NULL, n_null = 100L, sim_params = list(), verbose = TRUE)
```

Arguments

`hg`	An MSKHypergraph object.
`n_null`	Number of null models to generate (default: 100).
`sim_params`	List of simulation parameters (dt, n_steps, beta).
`verbose`	Logical, print progress.

Value

A list with:

null_scores Matrix of impact scores (n_muscles x n_null)

null_degree_scores List, impact scores grouped by degree for each null

n_null Number of null models

References

Murphy AC et al. (2018) PLOS Biology.

`mskNullHypergraph` *Generate Null Hypergraph*

Description

Creates a randomized version of the musculoskeletal hypergraph by rewiring muscle-bone connections while preserving each muscle's degree (number of bones it connects to). This is done by randomly reassigning which bones each muscle attaches to, within anatomical categories if specified.

Usage

```
mskNullHypergraph(hg = NULL, preserve_category = FALSE)
```

Arguments

`hg`	An MSKHypergraph object. If NULL, loads built-in data.
`preserve_category`	Logical, whether to preserve anatomical category during rewiring (default: FALSE for the basic model).

Value

An MSKHypergraph object with rewired connections.

References

Murphy AC et al. (2018) PLOS Biology.

`mskOutcomeConfidenceInterval` *Confidence Intervals for Outcome Predictions*

Description

Computes confidence intervals for functional outcome predictions using bootstrap resampling or analytical delta method.

Usage

```
mskOutcomeConfidenceInterval(
  predictions,
  method = c("bootstrap", "analytical"),
  n_boot = 100L,
  confidence_level = 0.95
)
```

Arguments

`predictions`	An `MSKFunctionalOutcome` object.
`method`	Character: "bootstrap" (default) or "analytical".
`n_boot`	Integer, number of bootstrap resamples (default: 100).
`confidence_level`	Numeric, confidence level (default: 0.95).

Value

A list with:

ci data.frame with outcome, point_estimate, lower, upper, width

method character

n_boot integer (for bootstrap)

overall_uncertainty numeric, mean CI width

Examples

```
## Not run:
outcome <- mskPredictFunctionalOutcome("Biceps Brachii")
ci <- mskOutcomeConfidenceInterval(outcome)

## End(Not run)
```

`mskOutcomeReport` *Comprehensive Outcome Report*

Description

Orchestrator function that combines functional outcome prediction, milestones, and optional reassessment into a comprehensive report.

Usage

```
mskOutcomeReport(outcome, milestones = NULL, reassessment = NULL, hg = NULL)
```

Arguments

`outcome`	An `MSKFunctionalOutcome` object.
`milestones`	Optional `MSKFunctionalMilestones` object.
`reassessment`	Optional `MSKReassessment` object.
`hg`	An MSKHypergraph object (NULL loads default).

Value

An S3 object of class `"MSKOutcomeReport"` with all sub-results.

Examples

```
## Not run:
outcome <- mskPredictFunctionalOutcome("Biceps Brachii")
milestones <- mskFunctionalMilestones(outcome)
report <- mskOutcomeReport(outcome, milestones)
print(report)

## End(Not run)
```

`mskOutcomeSummary` *MSK Outcome Summary Report*

Description

Orchestrator function that calls all clinical bridge functions with shared hypergraph and simulation objects, producing a comprehensive report.

Usage

```
mskOutcomeSummary(injury_muscles, patient_data = NULL, hg = NULL, sim = NULL)
```

Arguments

`injury_muscles`	Character or integer vector identifying injured muscles.
`patient_data`	Optional list with patient characteristics (see `mskInjuryRiskProfile`).
`hg`	An MSKHypergraph object (NULL loads default).
`sim`	An MSKSimulation object (NULL creates default).

Value

An S3 object of class `"MSKOutcomeSummary"` containing:

prediction MSKClinicalPrediction object

timeline Recovery timeline data.frame

risk_profile MSKInjuryRiskProfile object (if patient_data provided)

rehab MSKRehabProtocol object

Clinical Validity

All predictions are model-based estimates using MSK network topology. Recovery model was validated on 14 aggregate muscle groups. Patient factors are heuristic. This is a research exploration tool, not a clinical diagnostic.

Examples

```
## Not run:
summary <- mskOutcomeSummary("Trapezius")
print(summary)

## End(Not run)
```

`mskPathwayQuery` *Query Anatomical Pathway Between Entities*

Description

Finds the shortest anatomical pathway between two entities (muscles, bones, nerves, etc.) in the knowledge graph. Useful for tracing innervation chains, biomechanical linkages, and anatomical relationships.

Usage

```
mskPathwayQuery(from, to, hub = NULL, max_depth = 5L)
```

Arguments

`from`	Character, the source entity name (e.g., "Biceps Brachii").
`to`	Character, the target entity name (e.g., "C5").
`hub`	A PhysioAnnotationHub object (NULL loads default).
`max_depth`	Integer, maximum BFS depth (default: 5).

Value

A list with:

path Character vector of entities along the path

predicates Character vector of relationship types between entities

depth Integer, path length

description Human-readable path description

found Logical, whether a path was found

Clinical Validity

Pathways reflect relationships encoded in the knowledge graph. The shortest path in the KG may not correspond to the most clinically relevant connection. Always verify pathway interpretations against anatomical references.

Examples

```
## Not run:
path <- mskPathwayQuery("Biceps Brachii", "C5")
cat(path$description, "\n")

## End(Not run)
```

`mskPredictFunctionalOutcome`

Predict Functional Outcome from MSK Network Analysis

Description

Predicts clinical functional outcomes (ROM, strength, composite function score) for injured muscles based on MSK network topology, impact analysis, and optional EMG activation data. Provides evidence-based regression models with confidence intervals.

Usage

```
mskPredictFunctionalOutcome(
  injured_muscles,
  hg = NULL,
  outcome_type = c("all", "rom", "strength", "function"),
  patient_factors = NULL,
  emg = NULL,
  emg_mapping = NULL,
  confidence_level = 0.95
)
```

Arguments

`injured_muscles`	Character or integer vector identifying injured muscles.
`hg`	An MSKHypergraph object (NULL loads default 173-bone/270-muscle network).
`outcome_type`	Character: "rom" (range of motion), "strength", "function" (composite functional score), or "all" (default).
`patient_factors`	Optional list with: age (numeric), sex (character), bmi (numeric), activity_level (character), injury_severity ("mild"/"moderate"/"severe").
`emg`	Optional EMG data (SummarizedExperiment, matrix, or numeric vector) for activation-based prediction refinement.
`emg_mapping`	Optional pre-computed data.frame from `emgToMSKMapping()`.
`confidence_level`	Numeric, confidence level for CIs (default: 0.95).

Value

An S3 object of class `"MSKFunctionalOutcome"` with:

predictions data.frame with muscle, outcome_type, predicted_value, lower_ci, upper_ci, unit

aggregate list with overall_rom, overall_strength, overall_function

recovery_weeks estimated weeks to reach 90 percent of predicted outcome

confidence_level numeric

model_type character

patient_factors_used list

Clinical Validity

The prediction models are based on MSK network topology and heuristic adjustments. They are not independently validated clinical tools. Use as a research exploration tool, not a clinical diagnostic.

Examples

```
## Not run:
outcome <- mskPredictFunctionalOutcome(
  c("Biceps Brachii", "Deltoid"),
  patient_factors = list(age = 55, injury_severity = "moderate")
)
print(outcome)

## End(Not run)
```

`mskReassess` *Reassess Patient Progress*

Description

Compares current physiological measurements to a previous assessment, computing progress metrics and generating recommendations.

Usage

```
mskReassess(
  current_data,
  previous_assessment,
  hg = NULL,
  emg_mapping = NULL,
  sr = NULL
)
```

Arguments

`current_data`	A list with: emg (required, matrix or SummarizedExperiment), and optional eeg, kinematics, force.
`previous_assessment`	A previous `MSKFunctionalOutcome` or `MSKReassessment` object.
`hg`	An MSKHypergraph object (NULL loads default).
`emg_mapping`	Optional pre-computed EMG-to-MSK mapping.
`sr`	Optional numeric sampling rate.

Value

An S3 object of class `"MSKReassessment"` with:

current_metrics data.frame of current activation/synergy metrics

previous_metrics data.frame from previous assessment

change data.frame with metric, previous, current, change, pct_change, significant

progress_summary character vector per metric

overall_progress weighted average progress score (0-100)

recommendation character

Examples

```
## Not run:
reassessment <- mskReassess(
  current_data = list(emg = emg_matrix),
  previous_assessment = outcome
)
print(reassessment)

## End(Not run)
```

`mskRecoveryTimeline` *Recovery Timeline with Phase Assignment*

Description

Generates a week-by-week recovery timeline based on exponential decay of network impact, with clinical phase assignments.

Usage

```
mskRecoveryTimeline(injury_muscles, hg = NULL, sim = NULL, n_weeks = 12L)
```

Arguments

`injury_muscles`	Character or integer vector identifying injured muscles.
`hg`	An MSKHypergraph object (NULL loads default).
`sim`	An MSKSimulation object (NULL creates default).
`n_weeks`	Integer, number of weeks to project (default: 12).

Value

A data.frame with columns: week, muscle, remaining_impact_pct, phase, milestone.

Clinical Validity

Phase assignments are based on exponential decay of network impact scores, not empirical clinical data. They should be interpreted as model-based estimates for research purposes only.

Examples

```
## Not run:
timeline <- mskRecoveryTimeline("Biceps Brachii")

## End(Not run)
```

`mskRecoveryTrajectoryFit` *Fit Recovery Trajectory to Longitudinal Activation Data*

Description

Fits parametric recovery curves (exponential, sigmoid, or linear) to longitudinal muscle activation data from an MSKLongitudinalTracker.

Usage

```
mskRecoveryTrajectoryFit(
  tracker,
  model = c("exponential", "sigmoid", "linear"),
  muscle_subset = NULL
)
```

Arguments

`tracker`	An `MSKLongitudinalTracker` object.
`model`	Character, recovery curve model: "exponential" (default), "sigmoid", or "linear".
`muscle_subset`	Optional character vector of muscle names to fit (NULL = all muscles).

Value

An S3 object of class `"MSKRecoveryTrajectory"` with:

fits list per muscle with coefficients, residuals, R-squared, AIC

predicted matrix of predicted values (muscles x timepoints)

model_type character

recovery_rate numeric vector (slope at midpoint for each muscle)

time_to_90pct estimated time to 90 percent of asymptotic recovery

Examples

```
## Not run:
traj <- mskRecoveryTrajectoryFit(tracker, model = "exponential")

## End(Not run)
```

`mskRehabProtocol` *Rehabilitation Protocol Based on Network Topology*

Description

Generates a phased rehabilitation protocol using MSK network community structure and shortest path distances to determine exercise progression.

Usage

```
mskRehabProtocol(injury_muscles, hg = NULL, n_phases = 3L)
```

Arguments

`injury_muscles`	Character or integer vector identifying injured muscles.
`hg`	An MSKHypergraph object (NULL loads default).
`n_phases`	Integer, number of rehabilitation phases (default: 3).

Value

An S3 object of class `"MSKRehabProtocol"` with per-phase muscle lists.

Clinical Validity

Phase ordering is based on network topology (community membership and shortest path distance), not validated rehabilitation protocols. Use as a research exploration tool, not clinical guidance.

Examples

```
## Not run:
protocol <- mskRehabProtocol("Biceps Brachii")
print(protocol)

## End(Not run)
```

`mskReproducePaper` *Reproduce Paper Results*

Description

Master function that runs the complete analysis pipeline and compares results to the paper's reported values. This is the main validation function.

Usage

```
mskReproducePaper(run_simulation = FALSE, verbose = TRUE)
```

Arguments

`run_simulation`	Logical, run the full simulation (slow, default: FALSE). If FALSE, uses validation data to reproduce statistical analyses.
`verbose`	Logical, print progress and comparison.

Value

A list with all results and paper comparisons.

`mskRobustRegression` *Robust Regression for MSK Data*

Description

Fits a robust linear model (using MASS::rlm if available, otherwise lm). Supports weighted regression as used in the paper.

Usage

```
mskRobustRegression(x, y, weights = NULL)
```

Arguments

`x`	Numeric vector of predictor values.
`y`	Numeric vector of response values.
`weights`	Optional numeric vector of weights.

Value

A list with:

coefficients Named vector (intercept, slope)

r_squared R-squared value

f_statistic F-statistic

p_value p-value for the regression

residuals Residual values

fitted Fitted values

model The fitted model object

`mskShortestPaths` *Shortest Path Distances*

Description

Computes shortest path distance matrix for the projected graph.

Usage

```
mskShortestPaths(hg, type = c("bone", "muscle"))
```

Arguments

`hg`	An MSKHypergraph object.
`type`	Character, "bone" or "muscle" projection.

Value

A numeric matrix of shortest path distances.

`mskSimulate` *Simulate MSK Network Dynamics*

Description

Runs a damped harmonic oscillator simulation on the musculoskeletal network. Each muscle is modeled as a spring connecting its attached bones, and each bone is a unit-mass point particle. The system is evolved under perturbation and the total displacement is computed as the impact score.

Usage

```
mskSimulate(
  hg = NULL,
  dt = 0.01,
  n_steps = 500L,
  beta = 1,
  perturbation_magnitude = 1
)
```

Arguments

`hg`	An MSKHypergraph object. If NULL, loads the built-in data.
`dt`	Time step for integration (default: 0.01).
`n_steps`	Number of time steps to integrate (default: 500).
`beta`	Damping coefficient (default: 1.0).
`perturbation_magnitude`	Magnitude of the 4th-dimension perturbation (default: 1.0).

Value

An S3 object of class "MSKSimulation" with:

hg The MSKHypergraph used

spring_constants Named vector of spring constants per muscle

params List of simulation parameters

References

Murphy AC et al. (2018) PLOS Biology 16(1): e2002811.

`mskSynergyChangeIndex` *Compute Synergy Change Index Between Two Timepoints*

Description

Compares two synergy decompositions by aligning synergy weight vectors and computing a change index reflecting structural reorganization.

Usage

```
mskSynergyChangeIndex(
  synergy_t0,
  synergy_t1,
  method = c("cosine", "correlation", "procrustes")
)
```

Arguments

`synergy_t0`	An `MSKNeuromechSynergy` object or a W matrix (muscles x synergies) at baseline.
`synergy_t1`	An `MSKNeuromechSynergy` object or a W matrix at follow-up.
`method`	Character, comparison method: "cosine" (default), "correlation", or "procrustes".

Value

A list with: global_change_index (0=identical, 1=maximally different), per_synergy_change, alignment (permutation used).

Examples

```
## Not run:
sci <- mskSynergyChangeIndex(synergy_t0, synergy_t1, method = "cosine")

## End(Not run)
```

`mskZRand` *Adjusted Rand Index (z-score)*

Description

Computes the z-scored Rand index between two partitions. Used to compare community structure with homunculus categories.

Usage

```
mskZRand(partition1, partition2)
```

Arguments

`partition1`	Integer vector of community assignments.
`partition2`	Integer vector of category assignments.

Value

Numeric z-Rand score. Values > 1.96 indicate significant similarity.

References

Traud et al. (2011) Physical Review E.

`neuromechCorticomuscularCoupling`

Corticomuscular Coherence in MSK Network Context

Description

Computes pairwise corticomuscular coherence (CMC) between EEG and EMG channels, maps EMG channels to MSK muscles, builds a CMC-based functional distance matrix, and compares it with the structural muscle adjacency via Mantel test.

Usage

```
neuromechCorticomuscularCoupling(
  eeg,
  emg,
  hg = NULL,
  freq_band = c(15, 35),
  eeg_channels = NULL,
  emg_mapping = NULL,
  sr_eeg = NULL,
  sr_emg = NULL,
  nperseg = 256L,
  n_perm = 999L
)
```

Arguments

`eeg`	EEG data: a SummarizedExperiment, numeric matrix (time x channels), or numeric vector.
`emg`	EMG data: a SummarizedExperiment, numeric matrix (time x channels), or numeric vector.
`hg`	An MSKHypergraph object (NULL loads default).
`freq_band`	Numeric vector of length 2, frequency band in Hz for CMC (default: c(15, 35) for beta range).
`eeg_channels`	Optional character vector of EEG channel names to use. If NULL, motor cortex channels are auto-selected via `.eegChannelLookup()`.
`emg_mapping`	Optional pre-computed data.frame from `emgToMSKMapping()`.
`sr_eeg`	Optional sampling rate for EEG (overrides detected value).
`sr_emg`	Optional sampling rate for EMG (overrides detected value).
`nperseg`	Integer, segment length for Welch's method (default: 256).
`n_perm`	Integer, number of permutations for Mantel test (default: 999).

Value

An S3 object of class `"MSKNeuromechCMC"` with:

cmc_matrix Coherence matrix (n_eeg x n_emg)

structural_matrix MSK muscle adjacency (matched subset)

emg_cmc_profile Per-muscle mean CMC across EEG channels

significant_pairs Data.frame of significant EEG-EMG pairs

mantel Mantel test result (correlation, p_value)

mapping EMG-to-muscle mapping used

Examples

```
## Not run:
result <- neuromechCorticomuscularCoupling(eeg_data, emg_data)

## End(Not run)
```

`neuromechDirectionalCoupling` *Directional Cortico-Muscular Coupling*

Description

Computes directional (causal) connectivity between EEG and EMG channels using Granger causality or transfer entropy, distinguishing descending (cortical->muscle) from ascending (proprioceptive) pathways.

Usage

```
neuromechDirectionalCoupling(
  eeg,
  emg,
  hg = NULL,
  method = c("granger", "transfer_entropy", "both"),
  max_order_ms = 50,
  lag_ms = 20,
  n_bins = NULL,
  eeg_channels = NULL,
  emg_mapping = NULL,
  sr_eeg = NULL,
  sr_emg = NULL,
  n_perm = 999L,
  alpha = 0.05
)
```

Arguments

`eeg`	EEG data: SummarizedExperiment, matrix (time x channels), or vector.
`emg`	EMG data: SummarizedExperiment, matrix (time x channels), or vector.
`hg`	An MSKHypergraph object (NULL loads default).
`method`	Character: "granger" (default), "transfer_entropy", or "both".
`max_order_ms`	Numeric, maximum lag in ms for Granger (default: 50).
`lag_ms`	Numeric, TE lag in ms (default: 20).
`n_bins`	Integer, bins for TE discretization (NULL for auto).
`eeg_channels`	Optional character vector of EEG channels to use.
`emg_mapping`	Optional pre-computed data.frame from `emgToMSKMapping()`.
`sr_eeg`	Optional sampling rate for EEG.
`sr_emg`	Optional sampling rate for EMG.
`n_perm`	Integer, permutations for Mantel test (default: 999).
`alpha`	Numeric, significance threshold (default: 0.05).

Value

An S3 object of class `"MSKNeuromechDirectional"` with:

descending Matrix (n_eeg x n_emg): EEG->EMG values

ascending Matrix (n_emg x n_eeg): EMG->EEG values

net_direction descending - t(ascending)

significant_descending Data.frame of significant descending pairs

significant_ascending Data.frame of significant ascending pairs

dominance_ratio Per-muscle mean(descending)/mean(ascending)

pathway_classification Data.frame classifying each pair

msk_correlation Mantel test result

method Method used

parameters List of parameters used

Examples

```
## Not run:
result <- neuromechDirectionalCoupling(eeg_data, emg_data, method = "granger")

## End(Not run)
```

`neuromechElectromechanicalDelay`

Electromechanical Delay via Cross-Correlation

Description

Computes the electromechanical delay (EMD) between EMG and kinematic signals for muscle-bone pairs connected in the MSK hypergraph. Optionally correlates EMD with MSK network distance.

Usage

```
neuromechElectromechanicalDelay(
  emg,
  kinematics,
  hg = NULL,
  emg_mapping = NULL,
  kin_mapping = NULL,
  sr = NULL,
  max_lag_ms = 200,
  window_sec = NULL,
  n_perm = 999L
)
```

Arguments

`emg`	EMG data: SummarizedExperiment, matrix (time x channels), or vector.
`kinematics`	Kinematic data: matrix (time x segments) or SummarizedExperiment.
`hg`	An MSKHypergraph object (NULL loads default).
`emg_mapping`	Optional pre-computed data.frame from `emgToMSKMapping()`.
`kin_mapping`	Optional pre-computed data.frame from `mocapToMSKMapping()` or `imuToMSKMapping()`.
`sr`	Optional sampling rate (overrides detected value).
`max_lag_ms`	Numeric, maximum lag in milliseconds (default: 200).
`window_sec`	Optional numeric, window size in seconds for sliding window EMD analysis (NULL for global only).
`n_perm`	Integer, number of permutations for correlation test (default: 999).

Value

A list with:

emd Data.frame with muscle, bone, emd_ms, peak_correlation per pair

network_distance Corresponding MSK shortest path distance

correlation Pearson correlation between EMD and network distance

p_value Permutation p-value for the correlation

time_varying Data.frame with per-window EMD (if window_sec set)

Examples

```
## Not run:
result <- neuromechElectromechanicalDelay(emg_data, kin_data)

## End(Not run)
```

`neuromechIntegratedVulnerability`

Integrated Neuromechanical Vulnerability

Description

Combines EMG activation, kinematic stress, and force data to compute a multi-source vulnerability score for each muscle. Propagates kinematic stress through the MSK incidence matrix and weights by impact deviation.

Usage

```
neuromechIntegratedVulnerability(
  emg,
  kinematics,
  force_data,
  hg = NULL,
  emg_mapping = NULL,
  kin_mapping = NULL,
  weights = c(1, 1, 1),
  use_proxy = TRUE
)
```

Arguments

`emg`	EMG data: SummarizedExperiment, matrix, or vector.
`kinematics`	Kinematic data: matrix (time x segments) or SummarizedExperiment.
`force_data`	Force data: named numeric vector, data.frame, or matrix.
`hg`	An MSKHypergraph object (NULL loads default).
`emg_mapping`	Optional pre-computed data.frame from `emgToMSKMapping()`.
`kin_mapping`	Optional pre-computed data.frame from `mocapToMSKMapping()`.
`weights`	Numeric vector of length 3, weights for EMG, kinematics, force components (default: c(1, 1, 1)).
`use_proxy`	Logical, use degree-based proxy for impact deviation (default: TRUE).

Value

A list with:

vulnerability Named numeric vector of muscle vulnerability scores

ranking Data.frame sorted by vulnerability (descending)

source_correlations Pairwise correlations between EMG, kin, force sources

Examples

```
## Not run:
result <- neuromechIntegratedVulnerability(emg, kinematics, force)

## End(Not run)
```

`neuromechJointTorque` *Joint Torque Modeling in MSK Context*

Description

Estimates joint torques from EMG activation and anatomical moment arms, computing per-muscle contributions, coactivation indices, and torque balance ratios across joints.

Usage

```
neuromechJointTorque(
  emg,
  hg = NULL,
  emg_mapping = NULL,
  moment_arm_table = NULL,
  activation_method = c("rms", "mean_rectified", "peak"),
  sr = NULL,
  joints = NULL,
  n_perm = 999L
)
```

Arguments

`emg`	EMG data: SummarizedExperiment, matrix (time x channels), or vector.
`hg`	An MSKHypergraph object (NULL loads default).
`emg_mapping`	Optional pre-computed data.frame from `emgToMSKMapping()`.
`moment_arm_table`	Optional custom data.frame with columns: muscle_name, joint_name, moment_arm_m, direction. Overrides the built-in lookup.
`activation_method`	Character, method for computing activation: "rms" (default), "mean_rectified", "peak".

`sr`	Optional sampling rate.
`joints`	Optional character vector of joint names to restrict analysis.
`n_perm`	Integer, number of permutations for Mantel test (default: 999).

Value

An S3 object of class `"MSKNeuromechTorque"` with:

per_muscle Data.frame: muscle, joint, activation, moment_arm, direction, torque_contribution

per_joint Data.frame: joint, net_torque, agonist_sum, antagonist_sum, coactivation_index, n_muscles

torque_balance Data.frame: joint, balance_ratio

msk_correlation Mantel test result

moment_arm_source Character: "lookup" or "custom"

Examples

```
## Not run:
result <- neuromechJointTorque(emg_data, hg = hg)

## End(Not run)
```

`neuromechMotorDriveTopography`
Motor Drive Topography

Description

Maps cortical drive (CMC), muscle activation (EMG RMS), and force output to the MSK hypergraph to compute per-muscle drive efficiency and aggregate per community.

Usage

```
neuromechMotorDriveTopography(
  eeg,
  emg,
  force_data,
  hg = NULL,
  freq_band = c(15, 35),
  gamma = 4.3,
  emg_mapping = NULL,
  sr = NULL
)
```

Arguments

`eeg`	EEG data: SummarizedExperiment, matrix, or vector.
`emg`	EMG data: SummarizedExperiment, matrix, or vector.
`force_data`	Force data: named numeric vector (per muscle), data.frame with columns (muscle, force), or matrix (time x channels for RMS).
`hg`	An MSKHypergraph object (NULL loads default).
`freq_band`	Numeric vector of length 2, CMC frequency band (default: c(15, 35)).
`gamma`	Numeric, resolution parameter for community detection (default: 4.3).
`emg_mapping`	Optional pre-computed data.frame from `emgToMSKMapping()`.
`sr`	Optional sampling rate.

Value

A list with:

per_muscle Data.frame with muscle, cortical_drive, emg_activation, force, drive_efficiency

per_community Data.frame with community, mean drive/activation/force

drive_force_correlation Pearson r between cortical drive and force

drive_force_p_value p-value for the correlation

Examples

```
## Not run:
result <- neuromechMotorDriveTopography(eeg, emg, force)

## End(Not run)
```

`neuromechMuscleSynergy` *Muscle Synergy Decomposition in MSK Context*

Description

Extracts muscle synergies from EMG data via NMF or PCA, maps them to MSK structural communities, and compares synergy-based partitioning with structural communities via z-Rand.

Usage

```
neuromechMuscleSynergy(
  emg,
  hg = NULL,
  emg_mapping = NULL,
  method = c("nmf", "pca"),
  n_synergies = 4L,
```

```
  auto_select = FALSE,
  vaf_threshold = 0.9,
  max_k = 10L,
  gamma = 4.3,
  n_perm = 999L,
  seed = NULL,
  sr = NULL
)
```

Arguments

`emg`	EMG data: SummarizedExperiment, matrix (time x channels), or vector.
`hg`	An MSKHypergraph object (NULL loads default).
`emg_mapping`	Optional pre-computed data.frame from `emgToMSKMapping()`.
`method`	Character, decomposition method: "nmf" (default) or "pca".
`n_synergies`	Integer, number of synergies to extract (default: 4).
`auto_select`	Logical, automatically select n_synergies via VAF criterion (default: FALSE).
`vaf_threshold`	Numeric, VAF threshold for auto selection (default: 0.90).
`max_k`	Integer, maximum k to try during auto selection (default: 10).
`gamma`	Numeric, resolution parameter for MSK community detection (default: 4.3).
`n_perm`	Integer, permutations for z-Rand (unused, reserved).
`seed`	Optional integer seed for NMF reproducibility.
`sr`	Optional sampling rate.

Value

An S3 object of class `"MSKNeuromechSynergy"` with:

W Synergy weight matrix (n_muscles x n_synergies)

H Activation coefficients (n_synergies x n_timepoints)

n_synergies Number of synergies extracted

vaf Variance accounted for

vaf_curve VAF curve if auto_select (else NULL)

method Decomposition method used

community_mapping Synergy-to-community enrichment data.frame

synergy_similarity Cosine similarity matrix (k x k)

community_synergy_zrand z-Rand comparing synergy vs structural

reconstruction_error Reconstruction error

Examples

```
## Not run:
result <- neuromechMuscleSynergy(emg_data, method = "nmf", n_synergies = 4)

## End(Not run)
```

`neuromechSummary` *Neuromechanics Summary*

Description

Orchestrator function that runs all available neuromechanics analyses with shared mapping, skipping analyses when inputs are NULL.

Usage

```
neuromechSummary(
  eeg = NULL,
  emg,
  kinematics = NULL,
  force_data = NULL,
  hg = NULL,
  freq_band = c(15, 35),
  gamma = 4.3,
  sr = NULL,
  n_synergies = 4L,
  synergy_method = "nmf"
)
```

Arguments

`eeg`	Optional EEG data (NULL to skip CMC and motor drive analyses).
`emg`	EMG data (required).
`kinematics`	Optional kinematic data (NULL to skip EMD and vulnerability).
`force_data`	Optional force data (NULL to skip motor drive and vulnerability).
`hg`	An MSKHypergraph object (NULL loads default).
`freq_band`	Numeric vector of length 2, CMC frequency band (default: c(15, 35)).
`gamma`	Numeric, resolution parameter (default: 4.3).
`sr`	Optional sampling rate.
`n_synergies`	Integer, number of synergies for muscle synergy analysis (default: 4).
`synergy_method`	Character, synergy decomposition method: "nmf" (default) or "pca".

Value

An S3 object of class `"MSKNeuromechSummary"` with:

cmc CMC result or NULL

emd EMD result or NULL

motor_drive Motor drive result or NULL

vulnerability Vulnerability result or NULL

torque Joint torque result or NULL

synergy Muscle synergy result or NULL

directional Directional coupling result or NULL

available_analyses Character vector of successfully computed analyses

Examples

```
## Not run:
summary <- neuromechSummary(eeg = eeg_data, emg = emg_data,
                            kinematics = kin_data, force_data = force)

## End(Not run)
```

`opensimCompareModels` *Compare Multiple OpenSim Models*

Description

Runs network analysis on multiple OpenSim models and performs statistical comparisons of their network properties.

Usage

```
opensimCompareModels(model_paths, names = NULL, gamma = 4.3)
```

Arguments

`model_paths`	Character vector of paths to .osim files.
`names`	Optional character vector of model names.
`gamma`	Resolution parameter for community detection (default: 4.3).

Value

An S3 object of class `"MSKModelComparison"` with:

analyses List of MSKOpenSimAnalysis objects

comparisons Data frame of pairwise comparison statistics

`opensimForceToImpact` *OpenSim Force-weighted Impact Analysis*

Description

Combines muscle force data from OpenSim simulations with MSK network impact scores to compute force-weighted vulnerability.

Usage

```
opensimForceToImpact(
  model_path,
  force_data,
  hg = NULL,
  method = c("weighted_simulation", "weighted_scores")
)
```

Arguments

`model_path`	Character, path to .osim file.
`force_data`	Force data as a data.frame (with muscle name column), named numeric vector, or path to an OpenSim .sto file.
`hg`	An optional pre-built MSKHypergraph (if NULL, built from model).
`method`	Character, weighting method: **"weighted_simulation"** Modify spring constants k_m = force_m / (deg_m - 1) **"weighted_scores"** Post-hoc weighted_impact = impact * force

Value

A list with:

weighted_impact_scores Force-weighted impact scores

force_weights Force values used for weighting

unweighted_scores Original (unweighted) impact scores

`opensimNetworkAnalysis`

Full Network Analysis on OpenSim Model

Description

Builds a hypergraph from an OpenSim model and runs complete network analysis including metrics, community detection, and optionally impact scoring.

Usage

```
opensimNetworkAnalysis(
  model = NULL,
  model_path = NULL,
  gamma = 4.3,
  run_simulation = TRUE
)
```

Arguments

`model`	An optional PhysioOpenSimModel object.
`model_path`	Character, path to .osim file.
`gamma`	Resolution parameter for community detection (default: 4.3).
`run_simulation`	Logical, whether to run full impact simulation (default: TRUE; set to FALSE for faster analysis).

Value

An S3 object of class `"MSKOpenSimAnalysis"` with:

hypergraph The MSKHypergraph object

metrics Network metrics from mskNetworkMetrics

communities Community detection results

impact_scores Impact scores (if run_simulation = TRUE)

`opensimToMSKHypergraph`

Build MSKHypergraph from OpenSim Model

Description

Parses an OpenSim .osim file to extract muscle-bone attachment points and constructs a subject-specific MSKHypergraph.

Usage

```
opensimToMSKHypergraph(model = NULL, model_path = NULL)
```

Arguments

`model`	An optional PhysioOpenSimModel object. If provided, the model's path is used and results are cross-validated.
`model_path`	Character, path to a .osim file. Required if `model` is NULL.

Value

An `MSKHypergraph` object with subject-specific anatomy.

Note

Requires the `xml2` package.

Examples

```
## Not run:
hg <- opensimToMSKHypergraph(model_path = "gait2392.osim")
print(hg)

## End(Not run)
```

`plotCommunityStructure`

Plot Community Structure

Description

Visualizes detected communities with size information.

Usage

```
plotCommunityStructure(community, ...)
```

Arguments

`community`	Result from mskCommunityDetect or mskConsensusPartition.
`...`	Additional arguments.

Value

Invisible NULL.

`plotDegreeDistribution` *Plot Degree Distribution*

Description

Plots the degree distribution of the hypergraph, comparing real and null model distributions. Reproduces Fig 2e.

Usage

```
plotDegreeDistribution(
  hg = NULL,
  type = c("muscle", "bone"),
  null_dist = NULL,
  log_scale = TRUE,
  ...
)
```

Arguments

`hg`	An MSKHypergraph object.
`type`	"muscle" or "bone" distribution.
`null_dist`	Optional numeric vector of null model degree counts.
`log_scale`	Logical, use log scale (default: TRUE for heavy-tail).
`...`	Additional arguments passed to barplot.

Value

Invisible NULL.

`plotHomunculus` *Plot Homunculus Correspondence*

Description

Recreates Fig 4b: deviation ratio vs homunculus area. Target: F(1,19) = 21.3, R^2 = 0.52

Usage

```
plotHomunculus(homunculus_data = NULL, ...)
```

Arguments

`homunculus_data`	Optional data.frame. If NULL, uses built-in data.
`...`	Additional arguments.

Value

Invisible the regression result.

`plotImpactVsDegree` *Plot Impact Score vs Degree*

Description

Recreates Fig 3a: scatter plot of impact scores by hyperedge degree. Target: R^2 = 0.45 for the relationship.

Usage

```
plotImpactVsDegree(impact_scores, hg = NULL, show_regression = TRUE, ...)
```

Arguments

`impact_scores`	Named numeric vector of impact scores.
`hg`	An MSKHypergraph object.
`show_regression`	Logical, overlay regression line.
`...`	Additional arguments.

Value

Invisible the regression result.

`plotImpactVsRecovery` *Plot Impact Deviation vs Recovery Time*

Description

Recreates Fig 3b: weighted regression of impact deviation vs clinical recovery time. Target: R^2 = 0.757.

Usage

```
plotImpactVsRecovery(recovery_data = NULL, ...)
```

Arguments

`recovery_data`	Optional data.frame. If NULL, uses built-in data.
`...`	Additional arguments.

Value

Invisible the regression result.

`plotMSKNetwork` *Plot MSK Network*

Description

Visualizes the musculoskeletal network with optional community coloring.

Usage

```
plotMSKNetwork(
  hg = NULL,
  type = c("muscle", "bone"),
  membership = NULL,
  layout = "fr",
  vertex_size = 3,
  ...
)
```

Arguments

`hg`	An MSKHypergraph object.
`type`	"bone" or "muscle" projection to plot.
`membership`	Optional community membership vector for coloring.
`layout`	Character, layout algorithm: "fr" (Fruchterman-Reingold), "circle", or "auto".
`vertex_size`	Numeric, base vertex size.
`...`	Additional arguments passed to plot.

Value

Invisible NULL. Produces a plot.

`projectBoneGraph` *Project to Bone-centric Graph*

Description

Creates the bone-bone weighted adjacency matrix A = t(C) %*% C where A[i,j] counts the number of muscles shared between bones i and j. This is the one-mode projection onto the bone (vertex) space.

Usage

```
projectBoneGraph(hg)
```

Arguments

`hg`	An MSKHypergraph object, or a sparse incidence matrix.

Value

A symmetric sparse Matrix (n_bones x n_bones)

References

Murphy AC et al. (2018) PLOS Biology.

`projectMuscleGraph` *Project to Muscle-centric Graph*

Description

Creates the muscle-muscle weighted adjacency matrix B = C %*% t(C) where B[i,j] counts the number of bones shared between muscles i and j. This is the one-mode projection onto the muscle (hyperedge) space.

Usage

```
projectMuscleGraph(hg)
```

Arguments

`hg`	An MSKHypergraph object, or a sparse incidence matrix.

Value

A symmetric sparse Matrix (n_muscles x n_muscles)

References

Murphy AC et al. (2018) PLOS Biology.

`vertexDegree` *Vertex (Bone) Degree*

Description

Returns the degree of each vertex (bone), i.e., the number of muscles attached to each bone.

Usage

```
vertexDegree(hg)
```

Arguments

`hg`	An MSKHypergraph object or incidence matrix.

Value

Named numeric vector of bone degrees.

Index