

Package: PhysioRehab (via r-universe)

October 3, 2026

Title ICF-Native Single-Case Clinical-Reasoning Tools for Rehabilitation

Version 0.5.2

Description Re-centres physiological measurement around the way rehabilitation reasons about the individual: International Classification of Functioning (ICF) semantics, individualised goal-referenced outcomes (Goal Attainment Scaling), single-case change detection that fuses quantitative signals with qualitative ICF qualifiers, a hypothesis-to-evidence reasoning scaffold that keeps the clinician in the loop, and an auto-drafted, ICF-structured progress note. Adapters ingest real signals from the ecosystem (markerless gait from 'PhysioMoCap', muscle fatigue from 'PhysioEMG'); the ICF layer resolves against 'PhysioAnnotationHub'; single-case statistics come from 'PhysioAppKit'. The worked instance is stroke hemiplegic gait; the framework is condition-agnostic. A research tool with synthetic examples, not a medical device: outputs are clinician-in-the-loop measurement, documentation and reasoning support, never diagnosis.

License MIT + file LICENSE

Encoding UTF-8

URL <https://github.com/x-biosignal/PhysioRehab>,
<https://x-biosignal.github.io/PhysioRehab>

BugReports <https://github.com/x-biosignal/PhysioRehab/issues>

Imports PhysioAppKit, stats

Suggests PhysioAnnotationHub, PhysioCore, PhysioEMG, PhysioMoCap, S4Vectors, SummarizedExperiment, bslib, jsonlite, knitr, ragg, rmarkdown, shiny, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

RoxygenNote 7.3.3

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

Date/Publication 2026-10-03 13:11:50 UTC

RemoteUrl <https://github.com/x-biosignal/PhysioRehab>

RemoteRef HEAD

RemoteSha dc8c18a36845be4868e62cd64eb2bd7b3bcbe691

Contents

add_icf_measure	3
add_performance_from_freeliving	4
add_session	5
add_session_from_emg	6
add_session_from_mocap	7
adl_coverage	8
adl_coverage_summary	8
capacity_performance_gap	9
construct_coverage	10
draft_progress_note	11
emg_fatigue_measures	12
evaluate_hypothesis	12
fall_risk_from_episode	13
fall_risk_index	14
icf_catalog	16
icf_change	16
icf_construct_trajectory	17
icf_for_measure	18
icf_qualifier_label	18
launch_rehab_app	19
mocap_gait_measures	19
new_icf_construct	20
new_rehab_episode	21
plot_trajectory	21
rehab_app	22
rehab_goal	23
rehab_hypothesis	23
rehab_workflow	24
sced_analyze	25
simulate_stroke_gait_case	26

Index	27
--------------	-----------

add_icf_measure	<i>Add a measure observation to an ICF construct</i>
-----------------	--

Description

Records one modality's view of the domain, tagged as 'capacity' (a standardised/clinic measure – an ordinal scale total, a Rasch person measure, a performance test) or 'performance' (a free-living / real-world measure). The value is mapped to the ICF 0-4 qualifier either directly ('qualifier') or from 'cutpoints' (direction-aware), so heterogeneous measures become comparable. 'value' and 'date' may be vectors to add a whole longitudinal stream at once.

Usage

```
add_icf_measure(
  construct,
  source,
  value,
  qualifier_type = c("capacity", "performance"),
  modality = c("scale", "rasch", "performance_test", "sensor", "self_report"),
  date = NA,
  unit = NA_character_,
  direction = c("higher_better", "higher_worse"),
  qualifier = NULL,
  cutpoints = NULL
)
```

Arguments

construct	an 'icf_construct'.
source	short measure id, e.g. "barthel", "steps_per_day".
value	numeric measure value(s).
qualifier_type	"capacity" or "performance".
modality	"scale", "rasch", "performance_test", "sensor" or "self_report".
date	measurement date(s) (Date or coercible); recycled to 'value'.
unit	optional measure unit.
direction	"higher_better" (default) or "higher_worse".
qualifier	optional explicit ICF qualifier(s) 0-4 (overrides 'cutpoints').
cutpoints	optional length-4 ascending cut-points mapping 'value' to the 0-4 qualifier.

Value

the updated 'icf_construct'.

See Also

[capacity_performance_gap()], [icf_construct_trajectory()]

Examples

```
cc <- new_icf_construct("d450")
# A clinic scale (capacity) and a free-living sensor metric (performance),
# each mapped to the ICF 0-4 qualifier via ascending cut-points.
cc <- add_icf_measure(cc, "barthel_walk", value = 10,
                     qualifier_type = "capacity", modality = "scale",
                     cutpoints = c(2, 5, 8, 12))
cc <- add_icf_measure(cc, "steps_per_day", value = 1800,
                     qualifier_type = "performance", modality = "sensor",
                     cutpoints = c(1000, 3000, 6000, 9000))

cc
```

```
add_performance_from_freeliving
```

Add a free-living performance metric from a summary to an ICF construct

Description

Convenience adapter: pulls one aggregate metric from a ‘PhysioWearable::summarizeFreeLiving()’ result and records it as the ‘performance’ qualifier of the construct. Works structurally, so PhysioWearable need not be attached.

Usage

```
add_performance_from_freeliving(
  construct,
  fl,
  metric,
  cutpoints,
  date = NA,
  source = metric
)
```

Arguments

construct	an ‘icf_construct’.
fl	a ‘freeliving_summary’ (or any list with an ‘aggregate’ element).
metric	aggregate field name, e.g. “mvpa_min” or “steps”.
cutpoints	length-4 ascending cut-points mapping the metric to the 0-4 qualifier (higher metric = better performance).
date	measurement date.
source	measure id recorded (defaults to ‘metric’).

Value

the updated ‘icf_construct’.

See Also

[add_icf_measure()]

Examples

```
cc <- new_icf_construct("d450")
# Any list with an `aggregate` element works; here one free-living metric.
fl <- list(aggregate = list(steps = 2400))
cc <- add_performance_from_freeliving(cc, fl, metric = "steps",
                                     cutpoints = c(1000, 3000, 6000, 9000))
cc
```

add_session	<i>Record one therapy session</i>
-------------	-----------------------------------

Description

Record one therapy session

Usage

```
add_session(
  ep,
  date,
  phase,
  gait_speed = NA,
  cadence = NA,
  symmetry = NA,
  measures = list(),
  icf = list(),
  gas = list(),
  observation = ""
)
```

Arguments

- | | |
|-------------------|---|
| ep | a 'rehab_episode'. |
| date | session date (Date or string). |
| phase | "baseline" or "intervention". |
| gait_speed | comfortable gait speed (m/s), the primary measure. |
| cadence, symmetry | secondary quantitative measures (optional). |
| measures | named list of additional quantitative measures (arbitrary columns), e.g. 'list(knee_rom = 47.2)' for signal-derived measures. |
| icf | named list of ICF qualifiers, e.g. 'list(b770 = 3, d450 = 3)'. |
| gas | named list of GAS ratings keyed by goal id, e.g. 'list(G1 = -1)'. |
| observation | clinician's free-text qualitative note. |

Value

the updated 'rehab_episode'.

Examples

```
ep <- new_rehab_episode("PT-01", "stroke, left hemiplegia")
ep <- add_session(ep, "2026-04-06", "baseline", gait_speed = 0.41,
                  icf = list(d450 = 3), gas = list(G1 = -1))
ep
```

add_session_from_emg	<i>Append a rehab session whose measures come from a real EMG recording</i>
----------------------	---

Description

Append a rehab session whose measures come from a real EMG recording

Usage

```
add_session_from_emg(ep, date, phase, emg_pe, observation = "")
```

Arguments

ep	a 'rehab_episode'.
date	session date (Date or string).
phase	"baseline" or "intervention".
emg_pe	a <code>PhysioExperiment</code> of raw EMG.
observation	optional clinician note (auto-filled if empty).

Value

the updated 'rehab_episode', with real 'mdf_slope' / 'emg_rms'.

Examples

```
## Not run:
ep <- new_rehab_episode("PT-01", "stroke, left hemiplegia")
emg_pe <- PhysioEMG::make_emg_fatigue()
ep <- add_session_from_emg(ep, "2026-04-06", "baseline", emg_pe)

## End(Not run)
```

`add_session_from_mocap`*Append a rehab session whose measures come from a real OpenPose recording*

Description

Append a rehab session whose measures come from a real OpenPose recording

Usage

```
add_session_from_mocap(  
  ep,  
  date,  
  phase,  
  openpose_dir,  
  side = "R",  
  observation = ""  
)
```

Arguments

<code>ep</code>	a 'rehab_episode'.
<code>date, phase</code>	session date and phase.
<code>openpose_dir</code>	directory of OpenPose JSON frames for this visit.
<code>side</code>	"R"/"L".
<code>observation</code>	optional clinician note (auto-filled if empty).

Value

the updated 'rehab_episode', with real 'knee_rom' / 'knee_smoothness'.

Examples

```
## Not run:  
ep <- new_rehab_episode("PT-01", "stroke, left hemiplegia")  
ep <- add_session_from_mocap(ep, "2026-04-06", "baseline",  
                             openpose_dir = "path/to/openpose_frames")  
  
## End(Not run)
```

adl_coverage	<i>ADL measurement-coverage registry</i>
--------------	--

Description

What the ecosystem can measure for each ADL: the ICF Activities & Participation code, the measurement modality ('scale', 'performance_test', 'kinematics', 'emg', 'dexterity', 'sensor_freeliving', 'sensor_har'), the ICF qualifier it informs ('capacity' for standardised/clinic measures, 'performance' for real-world ones), and the function that provides it.

Usage

```
adl_coverage(code = NULL, qualifier = NULL)
```

Arguments

- code Optional ICF code(s) to filter to.
- qualifier Optional "capacity"/"performance" filter.

Value

a data.frame with 'code', 'title', 'modality', 'qualifier', 'method', 'package' and 'status' ("validated" = proven on real data or an established/verified measure; "supported" = pipeline + ICF mapping ready but not yet validated on a task-specific dataset).

See Also

```
[adl_coverage_summary()], [construct_coverage()]
```

Examples

```
adl_coverage("d640")
```

adl_coverage_summary	<i>Per-ADL coverage completeness</i>
----------------------	--------------------------------------

Description

Summarises, for each ADL code in the registry, how many modalities measure it and whether it is covered on the capacity and the performance qualifier – and hence whether it is genuinely multi-modal.

Usage

```
adl_coverage_summary()
```

Value

a data.frame 'code', 'title', 'n_modalities', 'has_capacity', 'has_performance', 'validated_performance' (at least one performance method proven on real data, not merely pipeline-ready) and 'multi-modal' (both qualifiers), with an attribute 'overall' giving ecosystem-level percentages including 'pct_validated_performance'.

See Also

[adl_coverage()]

Examples

```
adl_coverage_summary()
```

capacity_performance_gap

Capacity-performance gap for an ICF construct

Description

The signature rehabilitation discrepancy: how much a person's real-world doing (performance) falls short of what they can do in a standardised setting (capacity), both on the ICF 0-4 qualifier. A positive gap means performance is worse than capacity – the person does less than they are able, pointing to environmental or behavioural barriers rather than impairment.

Usage

```
capacity_performance_gap(construct, when = c("latest", "mean"))
```

Arguments

construct an 'icf_construct' carrying both a capacity and a performance qualifier.
 when "latest" (default, most recent of each) or "mean".

Value

a list: 'code', 'title', 'capacity', 'performance', 'gap' (performance - capacity) and a text 'interpretation'.

See Also

[add_icf_measure()]

Examples

```
cc <- new_icf_construct("d450")
cc <- add_icf_measure(cc, "barthel_walk", 10, "capacity", "scale",
  cutpoints = c(2, 5, 8, 12))
cc <- add_icf_measure(cc, "steps_per_day", 1800, "performance", "sensor",
  cutpoints = c(1000, 3000, 6000, 9000))
capacity_performance_gap(cc)
```

construct_coverage	<i>Coverage of a patient's ICF construct against the registry</i>
--------------------	---

Description

For the domain of one [new_icf_construct], reports which measurement modalities the ecosystem offers (from [adl_coverage()]) and which qualifiers the construct has actually populated, flagging the gaps – the modalities that are available but not yet used for this individual.

Usage

```
construct_coverage(construct)
```

Arguments

construct an 'icf_construct'.

Value

a list: 'code', 'title', 'has_capacity'/'has_performance' (in the construct), 'available' (registry rows for the code) and 'missing_qualifiers' (registry qualifiers with an available method but no observation yet).

See Also

[adl_coverage()], [capacity_performance_gap()]

Examples

```
cc <- new_icf_construct("d450")
cc <- add_icf_measure(cc, "barthel_walk", 10, "capacity", "scale",
  cutpoints = c(2, 5, 8, 12))
construct_coverage(cc)$missing_qualifiers
```

draft_progress_note	<i>Draft an ICF-structured progress note from an analysed episode</i>
---------------------	---

Description

Draft an ICF-structured progress note from an analysed episode

Usage

```
draft_progress_note(  
  ep,  
  sced,  
  reasoning = NULL,  
  mcid = sced$mcid,  
  constructs = NULL,  
  fall_risk = NULL  
)
```

Arguments

ep	a 'rehab_episode'.
sced	result of [sced_analyze].
reasoning	optional result of [evaluate_hypothesis].
mcid	MCID used (for the record).
constructs	optional list of [new_icf_construct] objects; when supplied, a cross-modal capacity-vs-performance section is added.
fall_risk	optional result of [fall_risk_from_episode()]; when supplied, an integrated fall-risk section is added.

Value

a single Markdown string (also the object printed in the demo).

Examples

```
ep <- simulate_stroke_gait_case()  
sced <- sced_analyze(ep)  
note <- draft_progress_note(ep, sced)  
cat(substr(note, 1, 160))
```

emg_fatigue_measures	<i>Real muscle-fatigue / amplitude measures from an EMG recording (via PhysioEMG)</i>
----------------------	---

Description

Real muscle-fatigue / amplitude measures from an EMG recording (via PhysioEMG)

Usage

```
emg_fatigue_measures(emg_pe, feature = c("mdf", "mnf"))
```

Arguments

emg_pe	a PhysioExperiment holding raw EMG (e.g. from PhysioIO, or the realistic ‘PhysioEMG::make_emg_fatigue()’ model).
feature	"mdf" (median) or "mnf" (mean) frequency.

Value

list: mdf_slope_hz_per_min, mdf_norm_slope_pct, fit_r2, rms.

Examples

```
## Not run:
# Requires PhysioEMG; emg_pe is a PhysioExperiment holding raw EMG.
emg_pe <- PhysioEMG::make_emg_fatigue()
emg_fatigue_measures(emg_pe)

## End(Not run)
```

evaluate_hypothesis	<i>Evaluate a hypothesis against the episode’s evidence</i>
---------------------	---

Description

Evaluate a hypothesis against the episode’s evidence

Usage

```
evaluate_hypothesis(ep, hyp, sced)
```

Arguments

ep	a ‘rehab_episode’.
hyp	a ‘rehab_hypothesis’.
sced	the result of [sced_analyze] on the primary measure.

Value

list(verdict, overall, items) where each item is one checked expectation with its supported/refuted status and the evidence used.

Examples

```
ep <- simulate_stroke_gait_case()
hyp <- rehab_hypothesis("task-oriented training improves gait",
  expect_icf = c(d450 = 1))
sced <- sced_analyze(ep)
evaluate_hypothesis(ep, hyp, sced)$overall
```

fall_risk_from_episode

Derive a fall-risk index from a rehab episode

Description

Builds [fall_risk_index()] domain inputs from a ‘rehab_episode’'s latest (or a chosen) session, mapping the standard domain names out of the session’s quantitative ‘measures’ columns (as stored by [add_session()]). Returns ‘NULL’ if the session holds none of the fall-risk domains, so it composes cleanly into [draft_progress_note()] / [rehab_workflow()].

Usage

```
fall_risk_from_episode(
  ep,
  session = NULL,
  reference = NULL,
  weights = NULL,
  cutpoints = NULL,
  strata_labels = NULL,
  domain_map = c(sway = "sway", gait_variability = "gait_variability", tug = "tug",
    strength = "strength", reaction_time = "reaction_time", vision = "vision",
    proprioception = "proprioception")
)
```

Arguments

ep	A ‘rehab_episode’ (see [new_rehab_episode()]).
session	Session number to use (default: the latest session).
reference, weights, cutpoints, strata_labels	Passed to [fall_risk_index()].
domain_map	Named character vector mapping fall-risk domains to session column names (default: identity for ‘sway’, ‘gait_variability’, ‘tug’, ‘strength’, ‘reaction_time’, ‘vision’, ‘proprioception’).

Value

A ‘fall_risk_index’, or ‘NULL’ if no domain columns are present.

See Also

[fall_risk_index()], [draft_progress_note()]

Examples

```
# Synthetic domain summaries recorded on one session (not clinical advice).
ep <- new_rehab_episode("PT-01", "stroke, left hemiplegia")
ep <- add_session(ep, "2026-04-06", "baseline",
  measures = list(sway = 12, tug = 18, strength = 38))
fall_risk_from_episode(ep)
```

fall_risk_index	<i>Integrated multi-domain fall-risk index</i>
-----------------	--

Description

Combines per-domain fall-risk summaries into a single index. Each argument is a scalar summary from the corresponding validated measure (e.g. ‘sway’ from [PhysioMoCap::swayMetrics()] path length, ‘gait_variability’ from a [PhysioMoCap::summarizeGaitParameters()] CV, ‘tug’ from [PhysioClinical::scoreTUG()], ‘strength’ from [PhysioClinical::scoreMRCSum()]). Omit a domain by leaving it ‘NA’.

Usage

```
fall_risk_index(
  sway = NA_real_,
  gait_variability = NA_real_,
  tug = NA_real_,
  strength = NA_real_,
  reaction_time = NA_real_,
  vision = NA_real_,
  proprioception = NA_real_,
  reference = NULL,
  weights = NULL,
  direction = NULL,
  cutpoints = NULL,
  strata_labels = NULL,
  method = c("zscore_composite", "ppa"),
  ppa_coef = NULL
)
```

Arguments

sway, gait_variability, tug, strength, reaction_time, vision, proprioception	Per-domain scalar summaries (any may be 'NA' to omit).
reference	A 'data.frame' of the same-named domain columns measured in a healthy reference sample; required for the "zscore_composite" method (each domain is z-standardised against it, sign-aligned so higher = more risk). Without it the composite is 'NA'.
weights	Optional named per-domain weights for the composite (default equal).
direction	Optional named '+1'/'-1' overrides of the default domain orientation ('+1' = higher is riskier).
cutpoints	Optional ascending composite cut-points defining risk strata; supply only from validated fall-outcome data. Default 'NULL' = uncalibrated (no stratum).
strata_labels	Optional labels for the 'length(cutpoints) + 1' strata.
method	"zscore_composite" (default) or "ppa" (needs 'ppa_coef').
ppa_coef	For 'method = "ppa"', a named numeric vector of the published PPA discriminant weights per domain (plus optional 'intercept'); not bundled.

Value

An S3 'fall_risk_index': 'domains', sign-aligned 'z', 'composite', 'percentile' (normal, when calibrated by a reference), 'stratum', 'method' and 'calibrated'.

References

Lord SR, Menz HB, Tiedemann A (2003). A physiological profile approach to falls risk assessment and prevention. **Phys Ther** 83:237-252.

See Also

[PhysioClinical::scoreTUG()], [PhysioClinical::scoreBerg()]

Examples

```
ref <- data.frame(sway = c(5, 6, 4, 5, 6), gait_variability = c(3, 4, 3, 5, 4),
  tug = c(8, 9, 7, 10, 8), strength = c(55, 52, 58, 50, 56))
fall_risk_index(sway = 12, gait_variability = 10, tug = 18, strength = 38,
  reference = ref)
```

icf_catalog	<i>ICF catalogue (stroke gait core-set subset)</i>
-------------	--

Description

ICF catalogue (stroke gait core-set subset)

Usage

```
icf_catalog()
```

Value

A data.frame of ICF codes: 'code', 'component', 'title_en', 'title_ja'.

Examples

```
icf_catalog()
```

icf_change	<i>ICF qualifier change across an episode (improvement = qualifier decreases)</i>
------------	---

Description

ICF qualifier change across an episode (improvement = qualifier decreases)

Usage

```
icf_change(ep, code)
```

Arguments

ep	a 'rehab_episode'.
code	ICF code, e.g. "d450".

Value

list(code, from, to, improved) where 'improved' > 0 means the qualifier fell (clinical improvement).
 NULL if the code was never recorded.

Examples

```
ep <- simulate_stroke_gait_case()
icf_change(ep, "d450")
```

`icf_construct_trajectory`*Time-ordered trajectory of one qualifier stream (feeds single-case analysis)*

Description

Time-ordered trajectory of one qualifier stream (feeds single-case analysis)

Usage

```
icf_construct_trajectory(  
  construct,  
  qualifier_type = c("capacity", "performance")  
)
```

Arguments

`construct` an 'icf_construct'.

`qualifier_type` "capacity" or "performance".

Value

a data.frame('date', 'value', 'qualifier') ordered by date, ready to pass to [PhysioAppKit::nonoverlap_analyze()] or the rehab SCED tools.

See Also

[capacity_performance_gap()]

Examples

```
cc <- new_icf_construct("d450")  
cc <- add_icf_measure(cc, "steps_per_day", value = c(1500, 2200, 3100),  
  qualifier_type = "performance", modality = "sensor",  
  date = as.Date("2026-04-06") + c(0, 7, 14),  
  cutpoints = c(1000, 3000, 6000, 9000))  
icf_construct_trajectory(cc, "performance")
```

icf_for_measure	<i>Map a measure to ICF codes via the real ontology (PhysioAnnotation-Hub)</i>
-----------------	--

Description

Uses PhysioAnnotationHub's ICF-linking rules (Cieza 2005) so a measure such as "gait_speed" resolves to "d450" from the maintained ontology, not a hard-coded table. Signal-derived measures that are not yet in the ontology (e.g. "knee_rom") return 'character(0)'; the fix is to register the mapping in PhysioAnnotationHub, not to hard-code it here.

Usage

```
icf_for_measure(measure)
```

Arguments

measure	a measure / instrument id, e.g. "gait_speed".
---------	---

Value

character vector of ICF codes (possibly empty).

Examples

```
# Resolves a measure to ICF code(s); "gait_speed" maps to walking (d450).
icf_for_measure("gait_speed")
```

icf_qualifier_label	<i>Label an ICF qualifier (0-4 severity)</i>
---------------------	--

Description

ICF body-function / activity qualifiers: 0 no problem .. 4 complete problem. Lower is better; improvement means the qualifier decreases.

Usage

```
icf_qualifier_label(q)
```

Arguments

q	integer qualifier 0..4 (vectorised).
---	--------------------------------------

Value

character labels.

Examples

```
icf_qualifier_label(0:4)
```

launch_rehab_app	<i>Launch the PhysioRehab GUI in a browser</i>
------------------	--

Description

Launch the PhysioRehab GUI in a browser

Usage

```
launch_rehab_app(...)
```

Arguments

... passed to [shiny::runApp].

Examples

```
## Not run:
launch_rehab_app()

## End(Not run)
```

mocap_gait_measures	<i>Real gait/kinematic measures from an OpenPose recording (via PhysioMoCap)</i>
---------------------	--

Description

Reads the COCO-18 OpenPose JSON frames, builds a PhysioExperiment, runs [PhysioMoCap::poseFix], and derives real knee-kinematic measures for one side.

Usage

```
mocap_gait_measures(openpose_dir, side = c("R", "L"), fps = 30)
```

Arguments

openpose_dir	directory of ‘openpose_data<N>.json’ frames (COCO-18 flat).
side	"R" or "L".
fps	frame rate (Hz).

Value

a list of real measures: knee_rom (deg), knee_smoothness_fixed and _raw (mean |2nd diff| of the knee angle, after / before poseFix), frames, flagged_fraction, leg_swaps.

Examples

```
## Not run:
# Requires PhysioMoCap and a directory of COCO-18 OpenPose JSON frames.
m <- mocap_gait_measures("path/to/openpose_frames", side = "R")
m$knee_rom

## End(Not run)
```

new_icf_construct	<i>Create a cross-modal ICF construct for one Activity/Participation domain</i>
-------------------	---

Description

Create a cross-modal ICF construct for one Activity/Participation domain

Usage

```
new_icf_construct(code, title = NULL)
```

Arguments

code	an ICF code, e.g. "d450" (walking) or "d550" (eating).
title	optional human title; resolved from the ontology (PhysioAnnotationHub, else the local catalogue) when omitted.

Value

an empty object of class 'icf_construct'.

See Also

[add_icf_measure()], [capacity_performance_gap()]

Examples

```
new_icf_construct("d450")
```

new_rehab_episode	<i>Create an empty rehabilitation episode (single-case)</i>
-------------------	---

Description

Create an empty rehabilitation episode (single-case)

Usage

```
new_rehab_episode(patient_id, condition, design = "AB", goals = list())
```

Arguments

patient_id	de-identified patient id.
condition	free-text condition, e.g. "stroke, left hemiplegia".
design	single-case design label, e.g. "AB".
goals	list of [rehab_goal] objects.

Value

an object of class 'rehab_episode'.

Examples

```
new_rehab_episode("PT-01", "stroke, left hemiplegia", design = "AB")
```

plot_trajectory	<i>Plot a single-case trajectory with phase line, 2-SD band and MCID target</i>
-----------------	---

Description

Plot a single-case trajectory with phase line, 2-SD band and MCID target

Usage

```
plot_trajectory(
  ep,
  measure = "gait_speed",
  mcid = 0.16,
  file = NULL,
  ylab = NULL
)
```

Arguments

ep	a 'rehab_episode'.
measure	primary measure column.
mcid	MCID (drawn as the target line above baseline mean).
file	optional PNG path; if NULL, draws to the current device.
ylab	y-axis label.

Value

(invisibly) the file path, or NULL.

Examples

```
ep <- simulate_stroke_gait_case()
# Draw to the current graphics device; pass `file` to save a PNG instead.
plot_trajectory(ep, mcid = 0.16)
```

rehab_app

Build the PhysioRehab clinician prototype GUI

Description

Build the PhysioRehab clinician prototype GUI

Usage

```
rehab_app()
```

Value

a Shiny app object (pass to [shiny::runApp] or print to launch).

Examples

```
## Not run:
# Requires shiny and bslib.
app <- rehab_app()
shiny::runApp(app)

## End(Not run)
```

rehab_goal	<i>Define a rehabilitation goal (ICF-linked, GAS-scaled)</i>
------------	--

Description

Define a rehabilitation goal (ICF-linked, GAS-scaled)

Usage

```
rehab_goal(id, statement, icf_codes, baseline, target, unit, gas = NULL)
```

Arguments

<code>id</code>	short identifier, e.g. "G1".
<code>statement</code>	free-text clinical goal (the patient's own words are ideal).
<code>icf_codes</code>	character vector of ICF codes this goal maps to.
<code>baseline, target</code>	numeric baseline and target on the primary measure.
<code>unit</code>	measurement unit, e.g. "m/s".
<code>gas</code>	optional named character vector describing GAS levels "-2","-1","0","+1","+2".

Value

an object of class 'rehab_goal'.

Examples

```
# Synthetic, goal-referenced target on a personal measure (not clinical advice).
rehab_goal("G1", "Walk 10 m indoors unaided",
  icf_codes = c("d450", "d4500"),
  baseline = 0.40, target = 0.70, unit = "m/s")
```

rehab_hypothesis	<i>State a clinical hypothesis about expected change</i>
------------------	--

Description

State a clinical hypothesis about expected change

Usage

```
rehab_hypothesis(statement, expect_speed_mcid = TRUE, expect_icf = c(d450 = 1))
```

Arguments

statement free-text clinical hypothesis.
 expect_speed_mcid logical: expect the primary measure to exceed MCID.
 expect_icf named integer vector of ICF codes -> minimum expected qualifier improvement, e.g. 'c(d450 = 1, b735 = 1)'.

Value

an object of class 'rehab_hypothesis'.

Examples

```
rehab_hypothesis("task-oriented training improves gait",
  expect_speed_mcid = TRUE, expect_icf = c(d450 = 1))
```

rehab_workflow	<i>Run the full single-case rehab workflow (guided, one call)</i>
----------------	---

Description

Run the full single-case rehab workflow (guided, one call)

Usage

```
rehab_workflow(
  ep,
  hypothesis = NULL,
  measure = "gait_speed",
  mcid = 0.16,
  direction = "increase",
  out_dir = tempdir(),
  fall_risk_reference = NULL
)
```

Arguments

ep a 'rehab_episode'.

hypothesis optional 'rehab_hypothesis'.

measure primary measure column.

mcid MCID for the primary measure.

direction improvement direction.

out_dir directory for the trajectory PNG and progress-note Markdown.

fall_risk_reference optional healthy-reference 'data.frame' passed to [fall_risk_from_episode()] to calibrate the integrated fall-risk composite.

Value

(invisibly) list(sced, reasoning, report, plot_file, report_file).

Examples

```
ep <- simulate_stroke_gait_case()
# Writes the trajectory PNG and progress note into out_dir (here tempdir()).
res <- rehab_workflow(ep, out_dir = tempdir())
cat(substr(res$report, 1, 140))
```

sced_analyze	Analyse an episode’s primary measure with SCED + MCID (via PhysioAppKit)
--------------	--

Description

Analyse an episode’s primary measure with SCED + MCID (via PhysioAppKit)

Usage

```
sced_analyze(
  ep,
  measure = "gait_speed",
  mcid = 0.16,
  direction = c("increase", "decrease")
)
```

Arguments

- ep a ‘rehab_episode’.
- measure column name of the primary quantitative measure.
- mcid minimal clinically important difference on that measure.
- direction improvement direction.

Value

a list with NAP, Tau, baseline mean, latest, MCID verdict, 2-SD band.

Examples

```
ep <- simulate_stroke_gait_case()
res <- sced_analyze(ep, measure = "gait_speed", mcid = 0.16)
res[c("nap", "delta", "exceeds_mcid")]
```

`simulate_stroke_gait_case`*Simulate a stroke hemiplegic-gait single-case episode*

Description

Simulate a stroke hemiplegic-gait single-case episode

Usage

```
simulate_stroke_gait_case(seed = 42)
```

Arguments

`seed` RNG seed for reproducibility.

Value

a populated 'rehab_episode'.

Examples

```
ep <- simulate_stroke_gait_case()  
ep
```

Index

[add_icf_measure](#), 3
[add_performance_from_freeliving](#), 4
[add_session](#), 5
[add_session_from_emg](#), 6
[add_session_from_mocap](#), 7
[adl_coverage](#), 8
[adl_coverage_summary](#), 8

[capacity_performance_gap](#), 9
[construct_coverage](#), 10

[draft_progress_note](#), 11

[emg_fatigue_measures](#), 12
[evaluate_hypothesis](#), 12

[fall_risk_from_episode](#), 13
[fall_risk_index](#), 14

[icf_catalog](#), 16
[icf_change](#), 16
[icf_construct_trajectory](#), 17
[icf_for_measure](#), 18
[icf_qualifier_label](#), 18

[launch_rehab_app](#), 19

[mocap_gait_measures](#), 19

[new_icf_construct](#), 20
[new_rehab_episode](#), 21

[plot_trajectory](#), 21

[rehab_app](#), 22
[rehab_goal](#), 23
[rehab_hypothesis](#), 23
[rehab_workflow](#), 24

[sced_analyze](#), 25
[simulate_stroke_gait_case](#), 26