Get started with healthiar

Hi there!

This vignette will 1) tell you about `healthiar`, 2) help you install the `healthiar` package, and 3) show you how to use `healthiar` with the help of examples.

NOTE: By using healthiar you agree to the terms of use and confirm you have read the disclaimer.

NOTE: the development of `healthiar` is still ongoing. Any feedback regarding bugs, unclear documentation, ... is welcome and highly appreciated. Please provide feedback via a GitHub issue.

How to install `healthiar`

Initial installation

Prerequisite: access to the BEST-COST GitHub repository with your GitHub account. If you don't have access yet, write to alberto.castrofernandez@swisstph.ch or axel.luyten@swisstph.ch and provide them with the email address of the GitHub account they should give access to.

For initial installation follow the detailed steps in the README of the BEST-COST GitHub repository. The main steps are:

1. Generate a GitHub personal access token (PAT)
2. Use your PAT to connect RStudio with your GitHub profile (to be done only once)
3. Install the (newest version) of `healhtiar` by running code below

```
credentials::set_github_pat() # Paste GitHub PAT if prompted

credentials::install_github(
  repo = "best-cost/best-cost_WPs",
  subdir = "/r_package/healthiar",
  ref = "HEAD", # By default "HEAD"; branch name to install package from
  force = TRUE,
  build_vignettes = TRUE
)

library(healthiar)
```

Updating `healthiar`

After the initial installation we recommend to regularly update `healthiar` by running the code above.

About `healthiar`

The main function family of `healthiar` is the attribute family, used to attribute health impacts to a risk factor, e.g. noise or air pollution. The core function `attribute_health` can be used for most (non-lifetable) assessments. For lifetable assessments the functions ending in `..._from_lifetable` are used.

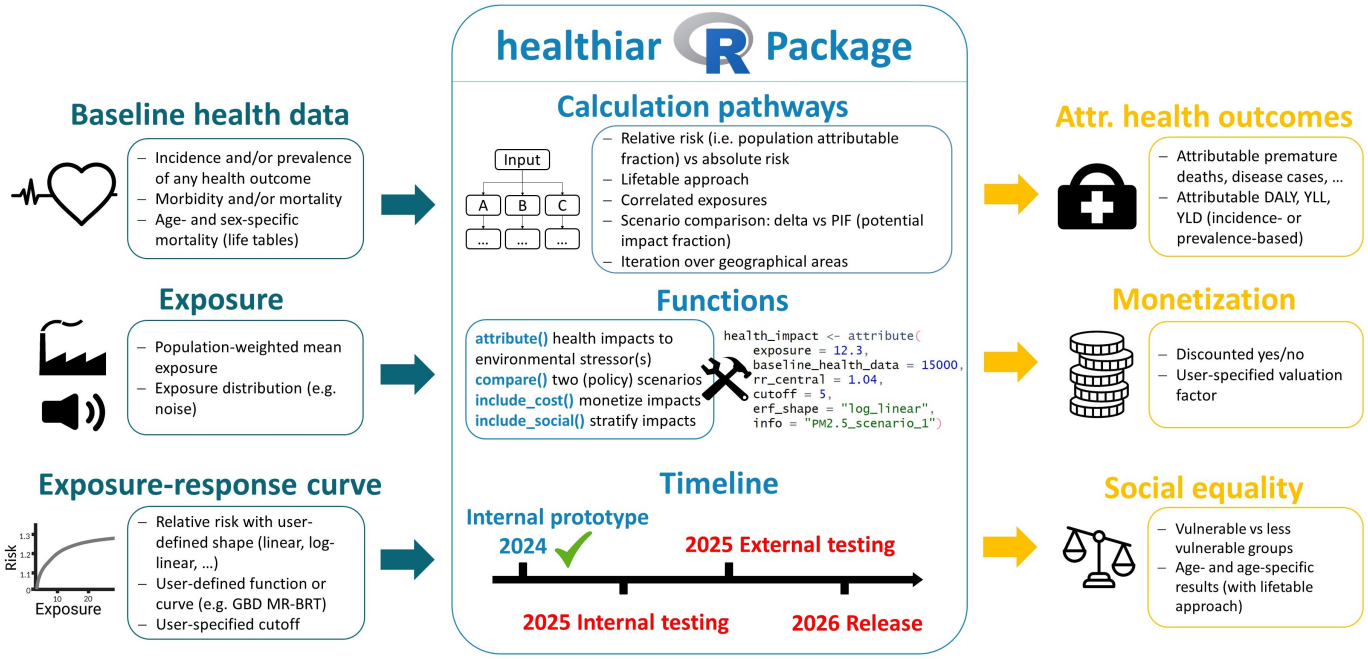

Figure 1: Figure: `healthiar` overview

`healthiar` comes with some example data that start with `exdat_` that allow you to test functions.

```
#> character(0)
```

How to use `healthiar`

Example relative risk: attribute COPD cases to pop-weighted mean exposure

Function call

We call the `attribute_health` with input data from the package example data. Note that we provide input data to the function argument using the `$` operator.

```
results_pm_copd <-
  healthiar::attribute_health(
    erf_shape = "log_linear",
    rr_central = exdat_pm_copd$relative_risk,
    rr_lower = exdat_pm_copd$relative_risk_lower,
    rr_upper = exdat_pm_copd$relative_risk_upper,
```

```
    rr_increment = 10,
    exp_central = exdat_pm_copd$mean_concentration,
    cutoff_central = exdat_pm_copd$cut_off_value,
    bhd_central = exdat_pm_copd$prevalence
  )
```

Results inspection

Every attribute output consists of two lists

- `health_main` contains the main results
- `health_detailed` detailed results (saved in the tibble `raw`) and (in some cases) even more information about the assessment/calculation

Let's inspect the main results

```
results_pm_copd[["health_main"]]
#> # A tibble: 3 x 22
#>   geo_id_disaggregated erf_ci  exp_ci  bhd_ci  cutoff_ci pop_fraction impact prop_pop_exp
#>   <chr>                <chr>   <chr>   <chr>   <chr>            <dbl>  <dbl>        <dbl>
#> 1 1                    central central central central         0.114   3502.            1
#> 2 1                    lower   central central central         0.0440  1353.            1
#> 3 1                    upper   central central central         0.178   5474.            1
#> # i 14 more variables: rr_increment <dbl>, erf_shape <chr>, exposure_name <lgl>,
#> #   approach_risk <chr>, health_outcome <chr>, exposure_dimension <int>, exposure_type <chr>,
#> #   exp <dbl>, rr <dbl>, bhd <dbl>, cutoff <dbl>, pop_fraction_type <chr>, rr_conc <dbl>,
#> #   impact_rounded <dbl>
```

It is a tibble of 3 rows and 22 columns. Let's zoom in on some relevant aspects

```
results_pm_copd |>
  pluck("health_main") |>
  select(exp, bhd, rr, erf_ci, pop_fraction, impact_rounded) |>
  knitr::kable() # Prints tibble in a minimal layout
```

exp	bhd	rr	erf_ci	pop_fraction	impact_rounded
8.85	30747	1.369	central	0.1138961	3502
8.85	30747	1.124	lower	0.0440064	1353
8.85	30747	1.664	upper	0.1780300	5474

Interpretation: this table shows us that exposure was 8.85 $\mu g/m^3$, the baseline health data (bhd) was 30747 (COPD cases in this instance). The 1st row further shows that the impact attributable to this exposure using the central relative risk (rr) estimate of 1.369 is 3502 COPD cases, or ~11% of all baseline cases.

The 2nd and 3rd rows show the impact using the lower and the upper estimates of the 95% confidence interval of the relative risk.

NOTE: the main output contains more columns that provide additional information about the assessment, such as cutoff, the relative risk at the observed exposure level, the shape of the ERF, ...

CONTINUE HERE: ADD HARD-CODED FUNCTION CALL FROM TESTING VIGNETTE

Including uncertainty in several input parameters

Now we will make a similar function call, but include uncertainty in several input arguments.

```
results_pm_copd <-
  healthiar::attribute_health(
    erf_shape = "log_linear",
    rr_central = exdat_pm_copd$relative_risk,
    rr_lower = exdat_pm_copd$relative_risk_lower,
    rr_upper = exdat_pm_copd$relative_risk_upper,
    rr_increment = 10,
    exp_central = exdat_pm_copd$mean_concentration,
    exp_lower = 8,
    exp_upper = 9,
    cutoff_central = exdat_pm_copd$cut_off_value,
    bhd_central = exdat_pm_copd$prevalence,
    bhd_lower = exdat_pm_copd$prevalence - 5000,
    bhd_upper = exdat_pm_copd$prevalence + 5000
    )
```

Let's inspect the detailed results:

```
results_pm_copd |>
  purrr::pluck("health_detailed") |>
  purrr::pluck("raw") |>
  dplyr::select(contains("_ci"), impact_rounded) |>
  knitr::kable() # Prints tibble in a minimal layout
```

erf_ci	exp_ci	bhd_ci	cutoff_ci	impact_rounded
central	central	central	central	3502
central	central	lower	central	2932
central	central	upper	central	4071
lower	central	central	central	1353
lower	central	lower	central	1133
lower	central	upper	central	1573
upper	central	central	central	5474
upper	central	lower	central	4584
upper	central	upper	central	6364
central	lower	central	central	2765

We see that each row represents a unique combination of the provided input variable estimates:

- The 1st row shows the impact when using the central estimates of each input variable
- The 2nd row shows the impact when using the central estimates of the relative risk, exposure in combination with the lower estimate of the baseline health
- ...

(*NOTE*: that only 9 of the 27 possible combinations are displayed due to space constraints.)

Adding summary uncertainty

You can do a Monte Carlo uncertainty analysis via the `include_summary_uncertainty` function.

```
results_pm_copd <-
  include_summary_uncertainty(
    results = results_pm_copd,
    n_sim = 1000
)
```

The outcome of the Monte Carlo analysis is added to the variable entered as the `results` argument, which is `results_pm_copd` in our case.

Two folders are added:

- `uncertainty_main` contains the central estimate and the corresponding 95% confidence intervals obtained through the Monte Carlo assessment
- `uncertainty_detailed` contains all `n_sim` simulations of the Monte Carlo assessment

```
print(
  results_pm_copd |>
  purrr::pluck("uncertainty_main")
  )
#> # A tibble: 1 x 3
#>   central_estimate lower_estimate upper_estimate
#>              <dbl>          <dbl>          <dbl>
#> 1            3418.          1185.          5543.

results_pm_copd |>
  purrr::pluck("uncertainty_detailed") |>
  purrr::pluck("raw") |>
  select(rr:impact_total) |>
  knitr::kable()
```

rr	rr_increment	erf_shape	exp	cutoff	bhd	dw	rr_conc	paf	impact_total
1.285688	10	log_linear	9.179196	5	32388.80	1	1.110734	0.0996941	3228.971
1.531341	10	log_linear	8.589651	5	29770.47	1	1.165291	0.1418452	4222.797
1.138642	10	log_linear	9.029772	5	25684.87	1	1.053714	0.0509758	1309.307
1.379912	10	log_linear	9.188967	5	31748.21	1	1.144414	0.1261907	4006.328
1.609170	10	log_linear	9.287921	5	31050.66	1	1.226279	0.1845249	5729.619
1.426400	10	log_linear	8.630718	5	28858.03	1	1.137629	0.1209788	3491.210
1.192879	10	log_linear	8.591441	5	32684.13	1	1.065391	0.0613776	2006.075
1.133884	10	log_linear	8.793815	5	27100.06	1	1.048823	0.0465505	1261.523
1.536306	10	log_linear	8.890516	5	28594.94	1	1.181815	0.1538438	4399.153
1.412146	10	log_linear	8.628246	5	31537.68	1	1.133392	0.1176924	3711.746

Adding monetization of results

You can monetize the obtained health impacts via the `include_monetization` function.

```
results_pm_copd <-
  include_monetization(
    output_healthiar = results_pm_copd,
    approach_discount = "direct",
    discount_shape = "exponential",
    discount_rate = 0.03,
    discount_years = 5,
    valuation = 50000, # E.g. EURO
    discount_overtime = "all_years"
  )

results_pm_copd |>
  purrr::pluck("monetization_main") |>
  select(erf_ci, impact, monetized_impact) |>
  knitr::kable()
```

erf_ci	impact	monetized_impact
central	3207.592	160379594
lower	1239.329	61966449
upper	5013.761	250688044

We see that the monetized impact (discounted) is more than 160 million EURO.

The outcome of the monetization is added to the variable entered to the `output_healthiar` argument, which is `results_pm_copd` in our case.

Two folders are added:

- `monetization_main` contains the central estimate and the corresponding 95% confidence intervals obtained through the specified monetization
- `monetization_detailed` contains the monetized results for each unique combination of the input variable estimates that were provided to the initial `attribute_health()` call

Adding a cost-benefit analysis (CBA) using the results

Let's imagine we design a policy that would reduce air pollution to 5 $\mu g/m^3$, which is the concentration specified in the `cutoff_central` argument in the initial `attribute_health()` call. So we could avoid all COPD cases attributed to air pollution.

What would be the monetary benefit of such a policy, considering also the cost to implement the policy (estimated at 100 million EURO)? We can find out using `healthiar`'s `include_cba()` function.

```
cba <-
  healthiar::include_cba(
    output_healthiar = results_pm_copd,
    approach_discount = "direct",
    valuation = 50000,
    cost = 100000000,
    discount_shape = "exponential",
    discount_rate_benefit = 0.03,
    discount_rate_cost = 0.03,
```

```
    discount_years_benefit = 5,
    discount_years_cost = 5,
    discount_overtime = "all_years"
  )

cba |>
  purrr::pluck("cba_main") |>
  select(benefit, cost, benefit_minus_cost) |>
  knitr::kable()
```

benefit	cost	benefit_minus_cost
160379594	91594144	68785451
61966449	91594144	-29627695
250688044	91594144	159093900

We see that the central and upper 95& confidence interval estimates of avoided attributable COPD cases result in a net monetary benefit of the policy, while the lower 95% confidence interval estimate results in a net cost!

The outcome of the CBA is contained in two folders are added:

- `cba_main` contains the central estimate and the corresponding 95% confidence intervals obtained
- `cba_detailed` contains additional intermediate results for both cost and benefit

Example absolute risk: attribute cases of high annoyance to (road) noise exposure

```
exdat_noise_ha <-
  exdat_noise_ha |>
  dplyr::filter(!is.na(exdat_noise_ha$exposure_mean))

results_noise_ha <-
  healthiar::attribute_health(
    approach_risk = "absolute_risk",
    exp_central = exdat_noise_ha$exposure_mean,
    population = sum(exdat_noise_ha$population_exposed_total),
    prop_pop_exp = exdat_noise_ha$population_exposed_total /
      sum(exdat_noise_ha$population_exposed_total),
    erf_eq_central = "78.9270-3.1162*c+0.0342*c^2")
```

erf_eq	erf_ci	impact_rounded
78.9270-3.1162c+0.0342c^2	central	174232

Example iteration: assess impact in multiple geographic units

To iterate an burden or impact assessment over multiple geo(graphic) units the geo-specific information, such as the numerical vectors entered as the `exp_...` and `bhd_...` arguments, have to be entered as a list, i.e. wrapped by the `as.list` function.

Additionally, specify the `geo_id_disaggregated` (e.g. "Brussels", "Antwerp", ...) and `geo_id_aggregated` (e.g. "Belgium") arguments. These IDs are used for aggregation of the results, i.e. all geo units that belong to the same administrative unit (Belgium in this case) have the same `geo_id_aggregated`.

(*NOTE*: in the example below the `runif` function from the `stats` package is used to simulate stroke cases and PM2.5 exposure for 5 Swiss cantons.)

```
results_iteration <-
  healthiar::attribute_health(
    exp_central = as.list(runif(5, 8.0, 9.0)),
    exp_lower = as.list(runif(5, 8.0, 9.0) - 0.1),
    exp_upper = as.list(runif(5, 8.0, 9.0) + 0.1),
    cutoff_central = 5,
    bhd_central = as.list(runif(5, 25000, 35000)),
    bhd_lower = as.list(runif(5, 25000, 35000) - 1000),
    bhd_upper = as.list(runif(5, 25000, 35000) + 1000),
    rr_central = 1.369,
    rr_lower = 1.124,
    rr_upper = 1.664,
    rr_increment = 10,
    erf_shape = "log_linear",
    geo_id_disaggregated = c("Zurich", "Basel", "Geneva", "Ticino", "Grisons"),
    geo_id_aggregated = rep("CH", 5), # CH = abbreviation of Switzerland
  )
```

Let's inspect the results from the iteration

```
results_iteration[["health_main"]] |>
  kable()
```

geo_id_aggregated	erf_ci	exp_ci	bhd_ci	exposure_name	impact	impact_rounded
CH	central	central	central	NA	14560.150	14560
CH	lower	central	central	NA	5605.924	5606
CH	upper	central	central	NA	22836.524	22837

We see that there are fewer columns than for a single geo unit assessment with `attribute()` because in this iteration case, the tibble `health_main` contains the *aggregated* results of the geo units. The cumulative / summed number of stroke cases attributable to PM2.5 exposure in the 5 geo units is 14560 (using a relative risk of 1.369).

The geo unit specific information and results are stored in `health_detailed`. Filter for the main result for each geo unit as follows

```
results_iteration[["health_detailed"]][["raw"]] |>
  pluck() |>
  ## Select main result for each geo unit
  ## i.e. rows where all ..._ci columns have value "central"
filter(
      if_all(.cols = matches("_ci"), .fns = ~ . == "central")
      ) |>
  select(geo_id_disaggregated, exp, bhd, rr, pop_fraction, impact_rounded) |>
  kable()
```

geo_id_disaggregated	exp	bhd	rr	pop_fraction	impact_rounded
Zurich	8.062515	28477.92	1.369	0.0917064	2612
Basel	8.107353	28730.30	1.369	0.0929847	2671
Geneva	8.854077	27686.50	1.369	0.1140095	3157
Ticino	8.387146	29750.33	1.369	0.1009204	3002
Grisons	8.877375	27195.11	1.369	0.1146576	3118

Besides the main result, `health_detailed` also contains impacts obtained through all combinations of input data central, lower and upper estimates.