

# Package ‘fairGATE’

May 8, 2026

**Title** Fair Gated Algorithm for Targeted Equity

**Version** 0.1.1

**Description** Tools for training and analysing fairness-aware gated neural networks for subgroup-aware prediction and interpretation in clinical datasets. Methods draw on prior work in mixture-of-experts neural networks by Jordan and Jacobs (1994) <[doi:10.1007/978-1-4471-2097-1\\_113](https://doi.org/10.1007/978-1-4471-2097-1_113)>, fairness-aware learning by Hardt, Price, and Srebro (2016) <[doi:10.48550/arXiv.1610.02413](https://doi.org/10.48550/arXiv.1610.02413)>, and personalised treatment prediction for depression by Iniesta, Stahl, and McGuffin (2016) <[doi:10.1016/j.jpsychires.2016.03.016](https://doi.org/10.1016/j.jpsychires.2016.03.016)>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Imports** dplyr, tibble, ggplot2, readr, pROC, magrittr, tidyr, purrr, utils, stats, ggalluvial, tidyselect, rlang

**Suggests** knitr, torch, testthat, readxl, rmarkdown

**VignetteBuilder** knitr

**URL** <https://github.com/rhysholland/FairGATE>

**BugReports** <https://github.com/rhysholland/FairGate/issues>

**Depends** R (>= 4.1.0)

**SystemRequirements** Optional 'LibTorch' backend; install via `torch::install_torch()`.

**LazyData** true

**NeedsCompilation** no

**Author** Rhys Holland [aut, cre],  
Raquel Iniesta [aut]

**Maintainer** Rhys Holland <[rhysh.holland@icloud.com](mailto:rhysh.holland@icloud.com)>

**Repository** CRAN

**Date/Publication** 2025-12-08 13:00:12 UTC

## Contents

adult_ready_small . . . . .	2
analyse_experts . . . . .	3
analyse_gnn_results . . . . .	4
export_f360_csv . . . . .	5
plot_sankey . . . . .	6
prepare_data . . . . .	7
train_gnn . . . . .	8

<b>Index</b>	<b>11</b>
--------------	-----------

---

adult_ready_small	<i>Adult ready (small sample)</i>
-------------------	-----------------------------------

---

### Description

A compact ( $n = 1000$ ) sample of the pre-cleaned Adult dataset for examples and vignettes. All original columns are retained; only rows are subsampled.

### Usage

```
adult_ready_small
```

### Format

A data frame with 1000 rows and K columns (same schema as the full dataset).

### Details

Created from the full CSV by random sampling with a fixed seed: `adult_ready_small <- dplyr::slice_sample(big, n = 1000)`.

### Source

UCI Adult (pre-cleaned in-package).

### Examples

```
data("adult_ready_small", package = "fairGATE")
str(adult_ready_small)
head(adult_ready_small)
```

**Description**

Analyses expert input weights to determine which features are most important per subgroup. For two groups, returns a signed difference plot (GroupB - GroupA). For >2 groups, performs a non-inferential analysis: per-group mean importances, pairwise (B - A) difference tables for all pairs, and a multi-group plot for the features with the largest max-min spread across groups.

**Usage**

```
analyse_experts(
  gnn_results,
  prepared_data,
  group_mappings = NULL,
  top_n_features = 10,
  verbose = FALSE
)
```

**Arguments**

`gnn_results` A list from `train_gnn()`.

`prepared_data` A list from `prepare_data()` (used to retrieve group mappings if not provided).

`group_mappings` Optional named mapping from group codes (names) to labels (values).

`top_n_features` Integer; number of top features to visualise.

`verbose` Logical; print progress messages.

**Value**

A list with:

`all_weights` Long table of feature importances by group & repeat means\_by\_group\_wide

`means_by_group_wide` Wide table of per-feature mean importance per group

`pairwise_differences` Named list of B\_vs\_A difference tables (descriptive)

`difference_plot` ggplot; only when there are exactly 2 groups

`multi_group_plot` ggplot; only when there are >2 groups

`top_features_multi` Long table used for the multi-group plot

---

analyse\_gnn\_results    *Analyse and Visualise GNN Results*

---

### Description

Generates ROC/Calibration outputs and subgroup gate analyses.

### Usage

```
analyse_gnn_results(
  gnn_results,
  prepared_data,
  group_mappings = NULL,
  create_roc_plot = TRUE,
  create_calibration_plot = TRUE,
  analyse_gate_weights = TRUE,
  analyse_gate_entropy = TRUE,
  verbose = FALSE,
  nonparametric = FALSE
)
```

### Arguments

gnn_results	List from train_gnn() (expects \$final_results, \$performance_summary, \$gate_weights).
prepared_data	List from prepare_data() (used to retrieve group mappings if not provided).
group_mappings	Optional named mapping from group codes (names) to labels (values).
create_roc_plot	Logical; return ROC ggplot (default TRUE).
create_calibration_plot	Logical; return calibration ggplot (default TRUE).
analyse_gate_weights	Logical; analyse gate weights across groups (default TRUE).
analyse_gate_entropy	Logical; analyse gate entropy across groups (default TRUE).
verbose	Logical; print progress messages.
nonparametric	Logical; if TRUE and k>2, use Kruskal-Wallis + Wilcoxon instead of ANOVA + Tukey.

### Value

A list of ggplots, test results, and summary tables.

---

export_f360_csv	<i>Export predictions for IBM Fairness 360</i>
-----------------	--

---

### Description

Create a CSV (or return a data frame) with columns expected by common IBM Fairness 360 workflows: subjectid, y\_true, y\_pred, score, group, group\_label, plus optional gate columns.

### Usage

```
export_f360_csv(
  gnn_results,
  prepared_data,
  path = NULL,
  include_gate_cols = TRUE,
  threshold = 0.5,
  verbose = FALSE
)
```

### Arguments

gnn_results	List returned by train_gnn(); must contain \$final_results.
prepared_data	List returned by prepare_data(); used to pull group mappings.
path	Optional file path to write the CSV. If NULL, no file is written.
include_gate_cols	Logical; include gate_prob_expert_* and gate_entropy if available. Default TRUE.
threshold	Numeric between 0 and 1 inclusive; classification threshold for y_pred. Default 0.5.
verbose	Logical; if TRUE, prints progress messages. Default FALSE.

### Value

Invisibly returns the data.frame that is written (or would be written).

### Examples

```
# Minimal toy example using simulated predictions
set.seed(1)

# Fake final_results: 20 subjects, binary outcome, probability scores
final_results <- data.frame(
  subjectid = 1:20,
  true      = sample(0:1, 20, replace = TRUE),
  prob      = runif(20),
  group     = sample(0:1, 20, replace = TRUE)
)
```

```

# gnn_results list in the shape returned by train_gnn()
gnn_results <- list(
  final_results = final_results
)

# prepared_data only needs group_mappings for labelling
prepared_data <- list()
attr(prepared_data, "group_mappings") <- c("0" = "Group 0", "1" = "Group 1")

# Write to a temporary CSV
tmp <- file.path(tempdir(), "fairness360_input.csv")
res <- export_f360_csv(
  gnn_results = gnn_results,
  prepared_data = prepared_data,
  path = tmp,
  include_gate_cols = FALSE,
  threshold = 0.5,
  verbose = FALSE
)

head(res)

```

---

plot\_sankey

---

*Create a Sankey Plot (robust; aggregates to one row per subject)*


---

### Description

Visualises routing from Actual Group -> Assigned Expert (2-axis), or Actual Group -> Learned Feature Profile -> Assigned Expert (3-axis) when feature mapping is available.

### Usage

```

plot_sankey(
  prepared_data,
  gnn_results,
  expert_results = NULL,
  top_n_per_side = 2,
  use_profiles = TRUE,
  verbose = FALSE
)

```

### Arguments

**prepared\_data** List from `prepare_data()`; used for group labels and (optionally) feature mapping. Needs `feature_names` for 3-axis; `subject_ids` to align subjects to X.

**gnn\_results** List from `train_gnn()`; uses `$final_results` and `$gate_weights`.

expert_results	Optional list from analyse_experts(); used only for picking opposed features in 3-axis.
top_n_per_side	Integer; number of features per side to define Profile A/B (default 2) for 3-axis.
use_profiles	Logical; try 3-axis when possible (default TRUE). If FALSE, always do 2-axis.
verbose	Logical; print progress.

**Value**

A ggplot object.

---

prepare_data	<i>Prepare Data for GNN Training</i>
--------------	--------------------------------------

---

**Description**

This function takes a raw dataframe, cleans it, defines the outcome and group variables, and scales the feature matrix. If no group\_mappings are provided, they are automatically generated from the unique values (or factor levels) of group\_var.

**Usage**

```
prepare_data(
  data,
  outcome_var,
  group_var,
  group_mappings = NULL,
  cols_to_remove = NULL
)
```

**Arguments**

data	A dataframe containing the raw data.
outcome_var	A string with the column name of the binary outcome (must be 0 or 1).
group_var	A string with the column name of the sensitive attribute.
group_mappings	Optional named list mapping values in group_var to numeric codes (e.g., list("Male" = 0, "Female" = 1)).
cols_to_remove	A character vector of column names to exclude from the feature matrix (e.g., IDs, highly collinear vars).

**Value**

A list containing:

X	The scaled feature matrix.
y	The numeric outcome vector.
group	The numeric group vector.
feature_names	The names of the features used.
subject_ids	A vector of subject IDs, if a 'subjectid' column exists.
group_mappings	Added as an attribute for downstream use.

**Examples**

```
my_data <- data.frame(
  subjectid = 1:10,
  remission = sample(0:1, 10, replace = TRUE),
  gender = sample(c("M", "F"), 10, replace = TRUE),
  feature1 = rnorm(10),
  feature2 = rnorm(10)
)

prepared_data <- prepare_data(
  data = my_data,
  outcome_var = "remission",
  group_var = "gender",
  cols_to_remove = c("subjectid")
)
```

---

train_gnn	<i>Train and Evaluate the Gated Neural Network (robust splits + safe ROC)</i>
-----------	---

---

**Description**

Trains a subgroup-aware gated neural network with a fairness-constrained loss, optionally performs hyperparameter tuning with lightweight budgets, and returns predictions, gate/expert weights, and summary metrics. Designed to be CRAN-safe:

- No background installs, no saving unless requested
- CPU-only torch, capped threads to avoid oversubscription

**Usage**

```
train_gnn(
  prepared_data,
  hyper_grid,
  num_repeats = 20,
  epochs = 300,
```

```

output_dir = tempdir(),
run_tuning = TRUE,
best_params = NULL,
save_outputs = FALSE,
seed = NULL,
verbose = FALSE,
tune_repeats = NULL,
tune_epochs = NULL
)

```

## Arguments

prepared_data	List from prepare_data() containing: <ul style="list-style-type: none"> <li>• X (matrix/data.frame of numeric features)</li> <li>• y (numeric 0/1)</li> <li>• group (numeric codes for sensitive subgroup)</li> <li>• feature_names (character vector; optional)</li> <li>• subject_ids (vector; optional)</li> </ul>
hyper_grid	data.frame with columns: lr, hidden_dim, dropout_rate, lambda, temperature.
num_repeats	Integer ( $\geq 1$ ). Repeated train/test splits for the <b>final</b> model (and for tuning if tune_repeats is not set).
epochs	Integer ( $\geq 1$ ). Training epochs per run for the <b>final</b> model (and for tuning if tune_epochs is not set).
output_dir	Directory to write csv/rds if save_outputs = TRUE. Defaults to tempdir().
run_tuning	Logical. If TRUE, runs a grid search using hyper_grid and picks best by mean AUC.
best_params	data.frame/list with lr, hidden_dim, dropout_rate, lambda, temperature if run_tuning = FALSE.
save_outputs	Logical. If TRUE, writes CSV/RDS outputs to output_dir. Default FALSE.
seed	Optional integer seed to make data splits reproducible. If NULL, current RNG state is respected.
verbose	Logical. Print progress messages. Default FALSE.
tune_repeats	Integer ( $\geq 1$ ). Repeats per combo <b>during tuning only</b> . Defaults to min(5, num_repeats).
tune_epochs	Integer ( $\geq 1$ ). Epochs per run <b>during tuning only</b> . Defaults to min(epochs, 100).

## Value

A list with:

- final\_results (tibble: subjectid, true, prob, group, iteration)
- gate\_weights (tibble with gate probabilities & entropy per subject/iteration)
- expert\_weights (list of expert input-layer weight matrices per repeat)

- `performance_summary` (tibble with AUC and Brier)
- `aif360_data` (tibble for fairness metric tooling)
- `tuning_results` (tibble or message when tuning skipped)

# Index

## \* datasets

adult\_ready\_small, [2](#)

adult\_ready\_small, [2](#)

analyse\_experts, [3](#)

analyse\_gnn\_results, [4](#)

export\_f360\_csv, [5](#)

plot\_sankey, [6](#)

prepare\_data, [7](#)

train\_gnn, [8](#)