Package: qvirus (via r-universe)

October 27, 2024

Title Quantum Computing for Analyzing CD4 Lymphocytes and Antiretroviral Therapy

Version 0.0.3

Description Resources, tutorials, and code snippets dedicated to exploring the intersection of quantum computing and artificial intelligence (AI) in the context of analyzing Cluster of Differentiation 4 (CD4) lymphocytes and optimizing antiretroviral therapy (ART) for human immunodeficiency virus (HIV). With the emergence of quantum artificial intelligence and the development of small-scale quantum computers, there's an unprecedented opportunity to revolutionize the understanding of HIV dynamics and treatment strategies. This project leverages the R package 'qsimulatR' (Ostmeyer and Urbach, 2023, <https://CRAN.R-project.org/package=qsimulatR>), a quantum computer simulator, to explore these applications in quantum computing techniques, addressing the challenges in studying CD4 lymphocytes and enhancing ART efficacy.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Suggests baguette, bookdown, earth, kknn, knitr, qsimulatR, rmarkdown, rules, testthat (>= 3.0.0), vdiffr, viralmodels, yardstick

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/juanv66x/qvirus

BugReports https://github.com/juanv66x/qvirus/issues

Depends R (>= 2.10)

LazyData true

Imports dplyr, factoextra, ggplot2, gridExtra, stats

NeedsCompilation no

Author Juan Pablo Acuña González [aut, cre, cph]
(<https://orcid.org/0009-0003-6029-6560>)

Maintainer Juan Pablo Acuña González <acua6307@gmail.com>

Repository CRAN

Date/Publication 2024-10-26 21:30:02 UTC

Contents

---

cd_3 *Longitudinal CD4 Lymphocyte Counts for HIV Patients (2018-2024)*

Description

Contains longitudinal measurements of CD4 lymphocyte counts for 176 patients living with HIV, recorded over the period from 2018 to 2024. CD4 counts are a critical indicator of immune function, used to monitor the progression of HIV and the effectiveness of treatments. Measurements were taken at various points throughout the study, with some missing values due to unavailable data for specific patients at certain times.

Usage

```
cd_3
```

Format

A data frame with 176 rows and 18 variables:

ID Unique identifier for each patient.

cd_2018_1 CD4 count for the first measurement in 2018.

cd_2018_2 CD4 count for the second measurement in 2018.

cd_2019_1 CD4 count for the first measurement in 2019.

cd_2019_2 CD4 count for the second measurement in 2019.

cd_2020_1 CD4 count for the first measurement in 2020.

cd_2021_1 CD4 count for the first measurement in 2021.

cd_2021_2 CD4 count for the second measurement in 2021.

cd_2021_3 CD4 count for the third measurement in 2021.

cd_2022_1 CD4 count for the first measurement in 2022.

cd_2022_2 CD4 count for the second measurement in 2022.

cd_2022_3 CD4 count for the third measurement in 2022.

cd_2023_1 CD4 count for the first measurement in 2023.

cd_2023_2 CD4 count for the second measurement in 2023.

cd_2023_3 CD4 count for the third measurement in 2023.

cd_2024_1 CD4 count for the first measurement in 2024.

cd_2024_2 CD4 count for the second measurement in 2024.

cd_2024_3 CD4 count for the third measurement in 2024.

Details

. CD4 counts are used to monitor immune system health in individuals with HIV. A lower CD4 count often indicates a weakened immune system, whereas higher counts suggest a stronger immune response. Some values are missing, indicating no measurement was taken for a particular patient at that time.

Source

Clinical data from Hospital Vicente Guerrero, IMSS, HIV Clinic.

Examples

```
# Load the dataset
data(cd_3)

# Summarize CD4 counts for the year 2021
summary(cd_3[, c("cd_2021_1", "cd_2021_2", "cd_2021_3")])
```

complex_check *Check if Coefficients of a Qubit State Object are Complex Numbers*

Description

This function returns the class of the coefficients of a given qubit state object

Usage

```
complex_check(qstate_obj)
```

Arguments

qstate_obj	A qubit state object created using the qsimulatR package.

Value

Coefficients class of given qubit state object

Examples

```
library(qsimulatR)
ket1 <- qstate(nbits = 1, coefs = c(0, 1))
complex_check(ket1)
```

conjugate_transpose *Calculate the Conjugate Transpose of a Quantum State*

Description

This function calculates the conjugate transpose of a given quantum state represented by a qstate object.

Usage

```
conjugate_transpose(state)
```

Arguments

state	A qstate object for which the conjugate transpose is to be calculated.

Value

The conjugate transpose of the input quantum state.

Examples

```
library(qsimulatR)
library(qvirus)
# Calculate the conjugate transpose of ket0
state <- six_state(1)[[1]]
conjugate_transpose(state)
```

`create_interactions` *Create Interactions from CD4 and Viral Load Data*

Description

This function computes the differences in CD4 counts and viral load from the provided datasets. It returns standardized values and logs transformed viral loads, enabling interaction analysis for further modeling.

Usage

```
create_interactions(cd_data, vl_data)
```

Arguments

`cd_data`	A data frame containing CD4 counts. Each row should represent an individual, and each column should represent a different time point or measurement.
`vl_data`	A data frame containing viral load measurements. It should have the same structure as `cd_data`, with rows for individuals and columns for time points.

Value

A list of class "interaction" containing:

cd_diff A numeric vector of CD4 differences computed from the input `cd_data`.

cds_diff A numeric vector of standardized CD4 differences.

vl_diff A numeric vector of viral load differences computed from the input `vl_data`.

vlog_diff A numeric vector of log-transformed viral load differences.

vlogs_diff A numeric vector of standardized log-transformed viral load differences.

Examples

```
# Example data frames for CD4 counts and viral loads
data(cd_3)
data(vl_3)

# Create interactions
create_interactions(cd_3[, -1], vl_3[, -1])
```

`estimate_payoffs` *Estimate Payoff Parameters for HIV Phenotype Interactions*

Description

This function estimates the payoff parameters for HIV phenotype interactions based on the provided classification object and predictions from a viral load model. It calculates the mean differences in viral loads and CD4 counts, as well as the average payoffs for each classification.

Usage

```
estimate_payoffs(class_obj, predictions_df)
```

Arguments

`class_obj`	An object of class `InteractionClassification` containing the data on viral load differences and CD4 counts.
`predictions_df`	A data frame containing predictions of viral loads, with a column named `predictions`. This data frame should align with the classifications in `class_obj`.

Value

A payoffs object containing the estimated payoff parameters

Examples

```
## Not run:
library(dplyr)
library(earth)
library(baguette)
library(rules)
library(kknn)
library(viralmodels)
# Load required data
data(vl_3)
data(cd_3)

# Create interaction object and classification
interaction_obj <- create_interactions(cd_3[,-1], vl_3[,-1])
class_obj <- InteractionClassification(interaction_obj$vlogs_diff, interaction_obj$cds_diff)

# Prepare predictions
traindata <- interaction_obj[c(2, 5)] |> as_tibble()
target <- "vlogs_diff"
predictions_df <- viralpreds(target, 2, 1, 2, 123, traindata)

# Estimate payoffs
payoffs_results <- estimate_payoffs(class_obj, predictions_df)

## End(Not run)
```

`InteractionClassification` *Interaction Classification for Viral Load and CD4 Differences*

Description

This function performs k-means clustering on viral load and CD4 count differences to classify interactions into distinct groups. It returns an S3 object containing the clustering results and means for each cluster.

Usage

```
InteractionClassification(vl_diff, cd_diff, k = 4, ns = 100, seed = 123)
```

Arguments

`vl_diff`	A numeric vector of viral load differences.
`cd_diff`	A numeric vector of CD4 count differences.
`k`	An integer specifying the number of clusters (default is 4).
`ns`	An integer specifying the number of random starts for the k-means algorithm (default is 100).
`seed`	An integer seed for reproducibility of the clustering results (default is 123).

Value

An S3 object of class `InteractionClassification`, containing:

`data`	A data frame with the original differences and their corresponding cluster classifications.
`kmeans_result`	The result of the k-means clustering, including cluster centers and within-cluster sum of squares.
`centers`	A matrix of the cluster centers.
`k`	The number of clusters used in the clustering.

Examples

```
data(vl_3)
data(cd_3)
interaction_obj <- create_interactions(cd_3[,-1], vl_3[,-1])
class_obj <- InteractionClassification(interaction_obj$vlogs_diff, interaction_obj$cds_diff)
```

mse *Mean Squared Errors for Interaction Classification*

Description

Mean squared errors (MSE) for viral load differences and CD4 count differences by comparing the actual values with the group means from the classification.

Usage

```
mse(object, ...)
```

Arguments

object	An object of class InteractionClassification containing the classified data and clustering results.
...	Additional arguments passed to other methods (currently not used).

Value

A list containing the MSE for viral load differences (vlogs_mse) and CD4 count differences (cds_mse).

Examples

```
data(vl_3)
data(cd_3)
interaction_obj <- create_interactions(cd_3[,-1], vl_3[,-1])
class_obj <- InteractionClassification(interaction_obj$vlogs_diff, interaction_obj$cds_diff)
mse(class_obj)
```

mse.InteractionClassification *Mean Squared Errors method for the InteractionClassification class*

Description

This method computes the mean squared errors (MSE) for viral load and CD4 differences based on the classification results from an InteractionClassification object.

Usage

```
## S3 method for class 'InteractionClassification'
mse(object, ...)
```

Arguments

object	An object of class InteractionClassification.
...	Additional arguments (currently not used).

---

nearest_payoff *Find Nearest Payoff*

Description

This function computes the nearest simulated payoff from a given list of payoffs based on a viral load difference (vl_diff). It returns both the nearest payoff value and its corresponding payoff name.

Usage

```
nearest_payoff(vl_diff, payoffs_list)
```

Arguments

vl_diff	Numeric value representing the viral load difference for which the nearest payoff will be found.
payoffs_list	A named list of payoff values, where the names correspond to specific payoffs and the values are the associated payoff values.

Value

A list with two elements: value (the nearest payoff value) and name (the name of the nearest payoff).

Examples

```
# Load data
library(dplyr)
library(qsimulatR)
data(vl_3)
data(cd_3)

# Create interaction object and classification
interaction_obj <- create_interactions(cd_3[,-1], vl_3[,-1])
class_obj <- InteractionClassification(interaction_obj$vlogs_diff, interaction_obj$cds_diff)

# Define gates and parameters for payoffs
gates <- list(
  T = Tgate(2),
  X = X(2),
  Id = Id(2),
  H = H(2),
  Z = Z(2),
  S = S(2),
```

```
  Y = Y(2)
)
alpha <- -0.4518303; beta <- -1.654192; gamma <- -0.2638399; theta <- -0.5619246
alpha2 <- -0.04186888; beta2 <- -3.01931; gamma2 <- 0.3922753; theta2 <-1.055114

# Generate the payoffs list
payoffs_list <- payoffs_list(gates, alpha, beta, gamma, theta, alpha2, beta2, gamma2, theta2)

# Filtered data based on specific patient IDs
filtered_data <- class_obj$data |>
  bind_cols(id = cd_3$ID) |>
  relocate(id) |>
  filter(id %in% c(37, 102, 148, 174, 180, 205))

# Apply the nearest_payoff function to each vl_diff in the filtered data
filtered_data <- filtered_data |>
  rowwise() |>
  mutate(nearest = list(nearest_payoff(vl_diff, payoffs_list))) |>
  mutate(nearest_payoff = nearest$value,
         payoff_name = nearest$name) |>
  select(-nearest) # Remove the intermediate column

# Display the updated filtered data with nearest payoff and its name
print(filtered_data)
```

`normalize_check` *Normalize Check Function for qstate Class Object*

Description

Check the normalization of a qstate object created by qsimulatR package.

Usage

```
normalize_check(qstate_obj, probs = FALSE)
```

Arguments

`qstate_obj`	A quantum state object.
`probs`	Are probabilities required as output?

Value

Either the sum of the squared magnitudes of the coefficients of the qstate object or its probabilities.

Examples

```
library(qsimulatR)
ket0 <- qstate(nbits = 1)
normalize_check(ket0)
```

payoffs_list *Generate Payoff List Based on Quantum Gates and Parameters*

Description

This function generates a list of payoffs for different combinations of quantum gate matrices. The payoffs are computed for two sets of parameters, where each set defines different values for the phenotype payoffs (v and V) in the quantum game model. The names of the payoffs are dynamically generated based on the provided gate names.

Usage

```
payoffs_list(gates, alpha, beta, gamma, theta, alpha2, beta2, gamma2, theta2)
```

Arguments

gates	A named list of gate matrices. Each element of the list is a quantum gate matrix (e.g., T, X, Id, H, Z, S, Y). The names of the list elements are used to create payoff names dynamically.
alpha	Numeric value for the first parameter set, defining payoff for v×v.
beta	Numeric value for the first parameter set, defining payoff for v×V.
gamma	Numeric value for the first parameter set, defining payoff for V×v.
theta	Numeric value for the first parameter set, defining payoff for V×V.
alpha2	Numeric value for the second parameter set, defining payoff for v×v.
beta2	Numeric value for the second parameter set, defining payoff for v×V.
gamma2	Numeric value for the second parameter set, defining payoff for V×v.
theta2	Numeric value for the second parameter set, defining payoff for V×V.

Value

A list of payoffs where the list names correspond to the gate combinations, and the values represent the computed payoffs based on the input parameters and gate matrices.

Examples

```
library(qsimulatR)
gates <- list(
  T = Tgate(2),
  X = X(2),
  Id = Id(2),
  H = H(2),
  Z = Z(2),
  S = S(2),
  Y = Y(2)
)
alpha <- 0.5; beta <- 0.2; gamma <- 0.3; theta <- 0.1
```

```
alpha2 <- 0.6; beta2 <- 0.25; gamma2 <- 0.35; theta2 <- 0.15
payoffs_list <- payoffs_list(gates, alpha, beta, gamma, theta, alpha2, beta2, gamma2, theta2)
print(payoffs_list)
```

---

phen_hiv *Calculate Final State and Payoffs in Quantum Game*

Description

This function calculates the final quantum state and expected payoffs for two players in a quantum game based on their strategies. The function uses quantum gates and unitary transformations to simulate the game dynamics.

Usage

```
phen_hiv(strategy1, strategy2, alpha, beta, gamma, theta)
```

Arguments

strategy1	A 2x2 matrix representing the strategy of player 1.
strategy2	A 2x2 matrix representing the strategy of player 2.
alpha	A numeric value representing the payoff for outcome |00>.
beta	A numeric value representing the payoff for outcome |01>.
gamma	A numeric value representing the payoff for outcome |10>.
theta	A numeric value representing the payoff for outcome |11>.

Value

A list containing the final quantum state (`final_state`), the payoffs for each basis state (`payoffs`), and the expected payoffs for player 1 (`pi_v`) and player 2 (`pi_V`).

References

Özlüer Başer, B. (2022). "Analyzing the competition of HIV-1 phenotypes with quantum game theory". Gazi University Journal of Science, 35(3), 1190–1198. doi:10.35378/gujs.772616

Examples

```
library(qsimulatR)
strategy1 <- diag(2) # Identity matrix for strategy 1
strategy2 <- diag(2) # Identity matrix for strategy 2
alpha <- 1
beta <- 0.5
gamma <- 2
theta <- 0.1
result <- phen_hiv(strategy1, strategy2, alpha, beta, gamma, theta)
print(result)
```

plot.interaction *Plot Interaction Differences*

Description

This function generates histograms and QQ plots for the differences in CD4 and viral load values contained in an interaction object. Users can choose to plot all differences or specify which type to plot.

Usage

```
## S3 method for class 'interaction'
plot(x, type = "all", ...)
```

Arguments

x	An interaction object containing the differences to be plotted. This object should include fields for CD4 differences (raw and standardized) and viral load differences (raw, log10-transformed, and log10-standardized).
type	A character string indicating the type of plot to generate. Options include: "cd_diff", "cds_diff", "vl_diff", "vlog_diff", "vlogs_diff", or "all" (the default) to plot all types of differences.
...	Additional arguments passed to the plot function.

Value

Plots histograms and QQ plots for the specified differences. If type is "all", all plots are arranged in a grid layout.

Examples

```
data("vl_3")
data("cd_3")
interaction_obj <- create_interactions(cd_3, vl_3)
plot(interaction_obj, type = "cd_diff")
```

plot.InteractionClassification *Plot InteractionClassification Clusters*

Description

This function visualizes the clusters formed by the InteractionClassification object. It displays the viral load differences (vl_diff) and CD4 count differences (cd_diff) as points, with each point color-coded by its cluster. Cluster centers are shown as red stars.

Usage

```
## S3 method for class 'InteractionClassification'
plot(
  x,
  plot_clusters = TRUE,
  evaluate_clusters = FALSE,
  max_clusters = 10,
  n_clusters = 4,
  ...
)
```

Arguments

x	An object of class InteractionClassification created using the InteractionClassification() function. It contains the clustering results and cluster centers.
plot_clusters	Logical. If TRUE, plots the cluster visualization. Default is TRUE.
evaluate_clusters	Logical. If TRUE, plots the WSS to evaluate the optimal number of clusters. Default is FALSE.
max_clusters	Integer. The maximum number of clusters to evaluate if evaluate_clusters is set to TRUE. Default is 10.
n_clusters	Integer. The number of clusters to suggest when plotting the elbow method. It determines where the vertical line (xintercept) is drawn in the plot. Default is 4.
...	Additional arguments (currently unused).

Details

Additionally, it can evaluate the optimal number of clusters using the elbow method by plotting the within-cluster sum of squares (WSS) using fviz_nbclust.

Value

A ggplot object that visualizes the clusters and their centers, or the WSS plot to evaluate the optimal number of clusters.

Examples

```
data(vl_3)
data(cd_3)
interaction_obj <- create_interactions(cd_3[,-1], vl_3[,-1])
class_obj <- InteractionClassification(interaction_obj$vlogs_diff, interaction_obj$cds_diff)
plot(class_obj)
```

print.interaction *Print Summary of CD4 and Viral Load Differences*

Description

This method prints a summary of the CD4 and viral load differences contained in the interaction object. It provides both raw and standardized values for CD4 differences, as well as raw, log10-transformed, and log10-standardized values for viral load differences.

Usage

```
## S3 method for class 'interaction'
print(x, ...)
```

Arguments

x	An object of class interaction containing CD4 and viral load differences.
...	Additional arguments passed to the summary function.

Value

This function does not return a value; it prints the summary statistics directly to the console.

Examples

```
data("vl_3")
data("cd_3")
interaction_obj <- create_interactions(cd_3, vl_3)
print(interaction_obj)
```

print.InteractionClassification *Print Method for InteractionClassification Objects*

Description

This method prints a summary of the InteractionClassification object, including the number of clusters, the cluster means, and the sizes of each cluster.

Usage

```
## S3 method for class 'InteractionClassification'
print(x, ...)
```

Arguments

x	An object of class InteractionClassification containing the results of the k-means clustering.
...	Additional arguments that may be passed to the print method.

Value

The function does not return a value; it prints the summary information to the console.

Examples

```
data(vl_3)
data(cd_3)
interaction_obj <- create_interactions(cd_3[,-1], vl_3[,-1])
class_obj <- InteractionClassification(interaction_obj$vlogs_diff, interaction_obj$cds_diff)

# Print the summary of the classification
print(class_obj)
```

---

pure_qubit1 *Create a normalized pure quantum state for a 1-qubit system.*

Description

Create a normalized pure quantum state for a 1-qubit system.

Usage

```
pure_qubit1(theta, phi, spherical = FALSE)
```

Arguments

theta	The parameter theta in radians.
phi	The parameter phi in radians.
spherical	Whether to return coordinates in spherical form (default is FALSE).

Value

A qstate object representing the normalized pure quantum state for a 1-qubit system.

Examples

```
# Quantum simulator
library(qsimulatR)
# Define the parameters
theta <- pi/4
phi <- pi/6
# Create the quantum state
psi_qubit1 <- pure_qubit1(theta, phi)
psi_qubit1
```

`pure_qubit2` *Create a normalized pure quantum state for a 2-qubit system.*

Description

Create a normalized pure quantum state for a 2-qubit system.

Usage

```
pure_qubit2(theta1, theta2, phi1, phi2)
```

Arguments

`theta1`	The parameter theta1 in radians for the first qubit.
`theta2`	The parameter theta2 in radians for the second qubit.
`phi1`	The phase parameter phi1 in radians for the first qubit.
`phi2`	The phase parameter phi2 in radians for the second qubit.

Value

A qstate object representing the normalized pure quantum state for a 2-qubit system.

Examples

```
#'
# Quantum simulator
library(qsimulatR)
# Define the parameters
theta1 <- pi/3
theta2 <- pi/4
phi1 <- pi/6
phi2 <- pi/5

# Create the quantum state
psi_qubit2 <- pure_qubit2(theta1, theta2, phi1, phi2)
psi_qubit2
```

simulate_entanglement *Simulate Entanglement Evolution*

Description

This function simulates the evolution of entanglement between two quantum states x1 and x2 using the CNOT gate.

Usage

```
simulate_entanglement(x1, x2, iterations, angle, verbose = FALSE)
```

Arguments

x1	Quantum state for qubit 1, represented as a qstate object.
x2	Quantum state for qubit 2, represented as a qstate object.
iterations	Number of iterations for the entanglement process.
angle	Rotation angle for applying Rx gate.
verbose	If TRUE, prints detailed information to the console.

Value

A list containing the entangled quantum state x2 after each iteration and other relevant information.

Examples

```
library(qsimulatR)
library(qvirus)
x1 <- qstate(1, coefs = as.complex(c(0.8, 0.6)))
x2 <- qstate(1, coefs = as.complex(c(0.38, 0.92)))
results <- simulate_entanglement(x1, x2, iterations = 3, angle = pi/4, verbose = TRUE)
print(results)
```

six_state *Create Six Important States on the Bloch Sphere*

Description

This function creates and returns six important states on the Bloch Sphere based on the specified vector numbers.

Usage

```
six_state(vec_num = c(1, 2, 3, 4, 5, 6))
```

Arguments

vec_num	A numeric vector specifying the indices of the states to include. Valid indices are 1 to 6.

Value

A list containing the selected quantum states based on the input vector vec_num.

Examples

```
library(qsimulatR)
# Select and return states 1, 3, and 5
six_state(c(1, 3, 5))
```

---

summary.interaction *Summary Method for Interaction Class Objects*

Description

This function provides a summary of various statistics for an interaction object, including raw and standardized differences for CD4 lymphocyte counts and viral loads. It returns a structured list of summary statistics, useful for understanding the differences in immune response and viral suppression between HIV phenotypes.

Usage

```
## S3 method for class 'interaction'
summary(object, ...)
```

Arguments

object	An object of class interaction that contains CD4 lymphocyte and viral load differences (raw and log-transformed).
...	Additional arguments (currently not used).

Value

A list with the following components:

cd_diff_raw Summary statistics for the raw differences in CD4 lymphocyte counts.

cd_diff_standardized Summary statistics for the standardized CD4 lymphocyte differences.

vl_diff_raw Summary statistics for the raw differences in viral loads.

vl_diff_log_transformed Summary statistics for the log-transformed viral load differences.

vl_diff_log_standardized Summary statistics for the standardized log-transformed viral load differences.

Examples

```
data(cd_3)
data(vl_3)
interaction_obj <- create_interactions(cd_3, vl_3)
summary(interaction_obj)
```

---

summary.InteractionClassification

Summarize Interaction Classification Results

Description

This function calculates and summarizes the mean viral load differences and CD4 count differences for each classification in the Interaction Classification object.

Usage

```
## S3 method for class 'InteractionClassification'
summary(object, ...)
```

Arguments

object	An object of class InteractionClassification containing the classified data and clustering results.
...	Additional arguments passed to other methods (currently not used).

Value

A data frame summarizing the mean viral load differences and CD4 count differences for each classification, along with the count of observations in each classification.

Examples

```
data(vl_3)
data(cd_3)
interaction_obj <- create_interactions(cd_3[,-1], vl_3[,-1])
class_obj <- InteractionClassification(interaction_obj$vlogs_diff, interaction_obj$cds_diff)
summary(class_obj)
```

summary.payoffs *Summarize Payoffs*

Description

This function summarizes the payoffs object by classification.

Usage

```
## S3 method for class 'payoffs'
summary(object, ...)
```

Arguments

object	A payoffs object.
...	Additional arguments (not used).

Value

A tibble summarizing the estimated payoffs.

Examples

```
## Not run:
library(dplyr)
library(earth)
library(baguette)
library(rules)
library(kknn)
library(viralmodels)
# Load required data
data(vl_3)
data(cd_3)

# Create interaction object and classification
interaction_obj <- create_interactions(cd_3[,-1], vl_3[,-1])
class_obj <- InteractionClassification(interaction_obj$vlogs_diff, interaction_obj$cds_diff)

# Prepare predictions
traindata <- interaction_obj[c(2, 5)] |> as_tibble()
target <- "vlogs_diff"
predictions_df <- viralpreds(target, 2, 1, 2, 123, traindata)

# Estimate payoffs
payoffs_results <- estimate_payoffs(class_obj, predictions_df)
summary(payoffs_results)

## End(Not run)
```

vl_3 *Longitudinal Viral Load Values for HIV Patients (2018-2024)*

Description

Contains longitudinal measurements of viral load for 176 patients from 2018 to 2024. Viral load is a critical marker used to monitor the effectiveness of HIV treatment by measuring the amount of HIV RNA in the blood.

Usage

```
vl_3
```

Format

A data frame with 176 rows and 18 variables:

ID Unique identifier for each patient.

vl_2018_1 Viral load for the first measurement in 2018.

vl_2018_2 Viral load for the second measurement in 2018.

vl_2019_1 Viral load for the first measurement in 2019.

vl_2019_2 Viral load for the second measurement in 2019.

vl_2020_1 Viral load for the first measurement in 2020.

vl_2021_1 Viral load for the first measurement in 2021.

vl_2021_2 Viral load for the second measurement in 2021.

vl_2021_3 Viral load for the third measurement in 2021.

vl_2022_1 Viral load for the first measurement in 2022.

vl_2022_2 Viral load for the second measurement in 2022.

vl_2022_3 Viral load for the third measurement in 2022.

vl_2023_1 Viral load for the first measurement in 2023.

vl_2023_2 Viral load for the second measurement in 2023.

vl_2023_3 Viral load for the third measurement in 2023.

vl_2024_1 Viral load for the first measurement in 2024.

vl_2024_2 Viral load for the second measurement in 2024.

vl_2024_3 Viral load for the third measurement in 2024.

Details

The viral load measurements provide insight into the patient's response to antiretroviral therapy (ART). Lower viral load values, especially undetectable levels, indicate better control of the infection. Missing values indicate that no viral load measurement was available for that patient at that specific time.

Source

Clinical data from Hospital Vicente Guerrero, IMSS, HIV Clinic.

Examples

```
## Not run:
# Load the dataset
data(vl_3)

# Summarize viral loads for the year 2021
summary(vl_3[, c("cd_2021_1", "cd_2021_2", "cd_2021_3")])

## End(Not run)
```

Index