Package: qvirus (via r-universe)

June 7, 2026

Title Quantum Computing for Analyzing CD4 Lymphocytes and Antiretroviral Therapy

Version 0.0.6

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 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

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 3.5)

LazyData true

Suggests rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports dplyr, magrittr, stats

Repository https://juan-acu.r-universe.dev

Date/Publication 2026-05-08 00:05:13 UTC

RemoteUrl https://github.com/juan-acu/qvirus

RemoteRef HEAD

RemoteSha 16722b7494f2f1ef28d37f20f01baaf97a885819

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")])
```

cd_diff *Create Mean Differences from Longitudinal CD4 Data*

Description

This function calculates the mean differences of CD4 counts across time for each individual in the dataset.

Usage

```
cd_diff(cd_data)
```

Arguments

cd_data	A data frame of longitudinal CD4 count values per individual, where rows represent patients and columns represent sequential measurements across time (e.g., years or visits).

Value

An object of class "Interaction" with the following components:

cd3_diff Mean differences of raw CD4 count values.

Examples

```
data(cd_3)
cd_data <- cd_3[,-1]
result <- cd_diff(cd_data)
```

cds_diff *Create Mean Standardized Differences from Longitudinal CD4 Data*

Description

This function calculates the mean standardized differences of CD4 counts across time for each individual in the dataset.

Usage

```
cds_diff(cd_data)
```

Arguments

cd_data	A data frame of longitudinal CD4 count values per individual, where rows represent patients and columns represent sequential measurements across time (e.g., years or visits).

Value

An object of class "Interaction" with the following components:

cds3_diff Mean standardized differences of raw CD4 count values.

Examples

```
data(cd_3)
cd_data <- cd_3[,-1]
result <- cds_diff(cd_data)
```

---

data_corr *Biomarker Correlation Dataset for Hamiltonian Construction*

Description

A numeric data frame containing a curated panel of biomarkers used to estimate correlation-driven coefficients for a variational Hamiltonian in a 3-qubit quantum model.

Usage

```
data(data_corr)
```

Format

A data frame with 12 rows and 30 variables:

ACMSD Amino acid metabolism biomarker.

ADA Nucleotide metabolism enzyme.

ADK Adenosine kinase (nucleotide metabolism).

CCL13 Chemokine involved in immune signaling.

CCL4 Inflammatory chemokine.

CCL5 Immune signaling chemokine.

CD14 Innate immune receptor.

CD163 Macrophage activation marker.

CPS1 Urea cycle enzyme.

CTSD Lysosomal protease.

CTSL Cathepsin L (antigen processing).

FAH Fumarylacetoacetate hydrolase.

FASN Fatty acid synthase.

HADH Beta-oxidation enzyme.

HK2 Hexokinase 2 (glycolysis).

IDO1 Tryptophan metabolism enzyme.

IL6 Inflammatory cytokine.

IMPDH2 Purine biosynthesis enzyme.

LAT T-cell activation linker protein.

LDHB Lactate dehydrogenase B.

LTC4S Leukotriene synthesis enzyme.

MTF1 Metal regulatory transcription factor.

NT5E Ecto-5'-nucleotidase.

PKM Pyruvate kinase (glycolysis).

PTGES Prostaglandin synthesis enzyme.

PTGS2 Cyclooxygenase-2 (inflammation).

RPTOR mTOR pathway regulator.

TDO2 Tryptophan metabolism enzyme.

ZNF708 Zinc finger transcription factor.

ZNF93 Zinc finger regulatory protein.

Details

The Hamiltonian is defined as:

$$\hat{H} = \sum_{k=1}^{K} c_k P_k$$

where:

- P_k are Pauli strings from the 3-qubit Pauli group
- c_k are coefficients estimated via Pearson correlation between selected biomarkers

The biomarkers in `data_corr` are grouped into biological axes:

- **Amino Acid Metabolism:** IDO1, TDO2, ACMSD, CPS1, FAH
- **Immune Signaling:** IL6, CCL4, CCL5, CCL13, NT5E
- **Antigen Processing:** CD14, CD163, LAT, CTSL, CTSD
- **Fatty Acid / Lipid:** HADH, FASN, LTC4S, PTGS2, PTGES
- **Aerobic / Nucleotide:** LDHB, PKM, IMPDH2, ADA, ADK, HK2
- **Regulatory / Epigenetic:** ZNF93, MTF1, RPTOR, ZNF708, CAD

These features define a high-dimensional correlation structure used to construct patient-specific or cohort-level Hamiltonians.

References

Brown, A. J., et al. (2024). Metabolic reprogramming of human macrophages during Mycobacterium tuberculosis infection under HIV exposure. Gene Expression Omnibus (GEO), accession GSE314344. Baluku, J. B., et al. (2023). DNA methylation profiling of people living with HIV stratified by tuberculosis status. Gene Expression Omnibus (GEO), accession GSE304107.

Examples

```
data(data_corr)
dim(data_corr)
```

`data_enc` *Angle-Encoding Dataset for 3-Qubit Quantum State Preparation*

Description

A numeric data frame containing patient-level biomarker expression values used for angle encoding into a 3-qubit quantum state. Each row corresponds to a sample (patient), and each column corresponds to a biomarker mapped to a rotation angle in the quantum circuit.

Usage

```
data(data_enc)
```

Format

A data frame with 12 rows and 6 variables:

CD14 Expression level of CD14 (innate immune receptor; phi2 angle).

HADH Expression level of HADH (beta-oxidation marker; theta3 angle).

IDO1 Expression level of IDO1 (tryptophan metabolism; phi1 angle).

IL6 Expression level of IL6 (inflammatory signaling; theta1 angle).

LDHB Expression level of LDHB (glycolytic flux; phi3 angle).

TDO2 Expression level of TDO2 (systemic metabolic repression; theta2 angle).

Details

The encoding follows a 3-qubit angle encoding scheme:

$$|\psi_j\rangle = \bigotimes_{i=1}^{3} R_y(\theta_{j,i}) R_z(\phi_{j,i}) |0\rangle$$

where each qubit is parameterized by two biomarkers:

- **Qubit 1:** `IL6` (θ_1), `IDO1` (ϕ_1)
- **Qubit 2:** `TDO2` (θ_2), `CD14` (ϕ_2)
- **Qubit 3:** `HADH` (θ_3), `LDHB` (ϕ_3)

These biomarkers capture key immunometabolic axes, including inflammation, tryptophan metabolism, innate immune sensing, and metabolic reprogramming.

Prior to encoding, values are typically normalized to angular ranges (e.g., $[0, 2\pi]$) using dataset-specific min–max scaling.

References

Brown, A. J., et al. (2024). Metabolic reprogramming of human macrophages during Mycobacterium tuberculosis infection under HIV exposure. Gene Expression Omnibus (GEO), accession GSE314344. Baluku, J. B., et al. (2023). DNA methylation profiling of people living with HIV stratified by tuberculosis status. Gene Expression Omnibus (GEO), accession GSE304107.

Examples

```
data(data_enc)
str(data_enc)
```

`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(object, predictions)
```

Arguments

`object`	An object of class `InteractionClassification` containing the data on viral load differences and CD4 counts.
`predictions`	A `data.frame` containing predictions of viral loads or CD4 values.

Examples

```
set.seed(42)
data(cd_3)
cd_data <- cd_3[,-1]
cd_result <- cds_diff(cd_data)
data(vl_3)
vl_data <- vl_3[,-1]
vl_result <- vlogs_diff(vl_data)
result <- InteractionClassification(cd_result = cd_result, vl_result = vl_result)
data(preds)
payoffs_results <- estimate_payoffs(result, preds)
```

`InteractionClassification` *Classify HIV phenotype interactions using k-means clustering*

Description

This function performs k-means clustering on the differences in viral load and CD4 counts to classify interaction types between HIV phenotypes. It returns an object of class `InteractionClassification`, a `data.frame` with classification labels.

Usage

```
InteractionClassification(cd_result, vl_result, k = 4, ns = 100, seed = 123)
```

Arguments

`cd_result`	A numeric vector of differences in CD4 T-cell counts.
`vl_result`	A numeric vector of differences in log viral load.
`k`	Integer. The number of clusters to use in k-means. Default is 4.
`ns`	Integer. Number of random initializations for the k-means algorithm. Default is 100.
`seed`	Integer. Seed for random number generation to ensure reproducibility. Default is 123.

Value

A `data.frame` of class `InteractionClassification` with three columns:

cds3_result The CD4 count difference for each interaction.

vlogs3_result The viral load difference (log scale) for each interaction.

classification An integer label (1 to `k`) indicating the interaction cluster.

Examples

```
set.seed(42)
data(cd_3)
cd_data <- cd_3[,-1]
cd_result <- cds_diff(cd_data)
data(vl_3)
vl_data <- vl_3[,-1]
vl_result <- vlogs_diff(vl_data)
result <- InteractionClassification(cd_result = cd_result, vl_result = vl_result)
```

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.

Computes the mean squared error (MSE) between observed CD4 and viral load differences and their corresponding predicted payoff values within each interaction classification.

Usage

```
mse(object, ...)

mse(object, ...)
```

Arguments

object	An object of class payoffs.
...	Additional arguments passed to other methods (currently not used).

Value

A data.frame containing the MSE for CD4 count differences (mse_cds_diff) and (mse_vlogs_diff) for viral load differences.

Examples

```
set.seed(42)
data(cd_3)
cd_data <- cd_3[,-1]
cd_result <- cds_diff(cd_data)
data(vl_3)
vl_data <- vl_3[,-1]
vl_result <- vlogs_diff(vl_data)
result <- InteractionClassification(cd_result = cd_result, vl_result = vl_result)
mse(result)

set.seed(42)
data(cd_3)
cd_data <- cd_3[,-1]
cd_result <- cds_diff(cd_data)
data(vl_3)
vl_data <- vl_3[,-1]
vl_result <- vlogs_diff(vl_data)
result <- InteractionClassification(cd_result = cd_result, vl_result = vl_result)
data(preds)
payoffs_results <- estimate_payoffs(result, preds)
```

```
mse(payoffs_results)
```

`mse.InteractionClassification` *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

```
## S3 method for class 'InteractionClassification'
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).

`mse.payoffs` *Mean Squared Errors for Payoff Predictions*

Description

Computes the mean squared error (MSE) between observed CD4 and viral load differences and their corresponding predicted payoff values within each interaction classification.

Usage

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

Arguments

`object`	An object of class `payoffs`.
`...`	Additional arguments passed to other methods (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.

Examples

```
I <- diag(2)
H <- 1 / sqrt(2) * matrix(c(1, 1, 1, -1), 2, 2)
Z <- diag(c(1, -1))
gates <- list(I = I, H = H, Z = Z)
alpha <- 0.3; beta <- 0.1; gamma <- 0.5; theta <- 0.2
alpha2 <- 0.35; beta2 <- 0.15; gamma2 <- 0.6; theta2 <- 0.25
pays <- payoffs_list(gates, alpha, beta, gamma, theta, alpha2, beta2, gamma2, theta2)
nearest_payoff(-0.2, pays)
```

payoffs_list *Compute Payoff Values for Quantum HIV Phenotype Interactions*

Description

Computes payoff values for all pairwise combinations of quantum gate strategies provided in a named list. For each pair, the function calculates the payoffs for both phenotypes v and V using two different sets of payoff parameters.

Usage

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

Arguments

gates	A named list of 2x2 unitary matrices representing quantum strategies (e.g., I, H, Z).
alpha	Numeric scalar, payoff coefficient for phenotype v when both play 0.
beta	Numeric scalar, payoff coefficient for phenotype v when v plays 0, V plays 1.
gamma	Numeric scalar, payoff coefficient for phenotype v when v plays 1, V plays 0.
theta	Numeric scalar, payoff coefficient for phenotype v and V when both play 1.
alpha2	Numeric scalar, alternate value of alpha for phenotype v in a second scenario.
beta2	Numeric scalar, alternate value of beta for phenotype v in a second scenario.
gamma2	Numeric scalar, alternate value of gamma for phenotype v in a second scenario.
theta2	Numeric scalar, alternate value of theta for phenotype v in a second scenario.

Examples

```
I <- diag(2)
H <- 1 / sqrt(2) * matrix(c(1, 1, 1, -1), 2, 2)
Z <- diag(c(1, -1))
gates <- list(I = I, H = H, Z = Z)
payoffs <- payoffs_list(gates, 1, 0.5, 0.3,0.2, 1.5, 0.6, 0.7, 0.8)
```

---

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>.

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

```
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)
```

---

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

Description

Contains predictions of longitudinal viral load values for 176 patients from 2018 to 2024.

Usage

```
preds
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 176 rows and 1 columns.

Source

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

Examples

```
data(preds)
head(preds)
```

preds2 — *Batched Predictions for Longitudinal Viral Load Values for HIV Patients (2018-2024)*

Description

Contains batched predictions of longitudinal viral load values for 176 patients from 2018 to 2024.

Usage

```
preds2
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 176 rows and 1 columns.

Source

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

Examples

```
data(preds2)
head(preds2)
```

qphen — *Quantum Phenotype Interactions in HIV Model*

Description

The qphen dataset contains 176 observations and 24 variables, representing classified phenotype interactions in a quantum game-theoretic model of HIV phenotypes. The data includes CD4 and viral load differences, quantum game strategies, classification clusters, and tuberculosis/genoresistance indicators.

Usage

```
data(qphen)
```

Format

A data frame with 176 rows and 24 variables:

id (double) Unique identifier for each observation.

vl_diff (double) Difference in viral load (log scale) between time points.

cd_diff (double) Difference in CD4 count between time points.

vlogs_diff_mean (double) Mean difference of viral loads across the dataset.

cds_diff_mean (double) Mean difference of CD4 counts across the dataset.

n (double) Number of cases in each interaction cluster.

payoffs (double) Computed payoff for the phenotype interaction. #'

payoffs_b (double) Alternative computed payoff.

nearest_payoff (double) Closest estimated payoff value.

classification_2 (double) Cluster assignment for phenotype interactions (second clustering method).

classification_3 (double) Cluster assignment for phenotype interactions (third clustering method).

classification_4 (double) Cluster assignment for phenotype interactions (fourth clustering method).

phen_1 (double) Phenotype type (`v` or `V`).

str1_2 (double) Strategy of the first phenotype using `X`, `T`, or `H` gate (binary encoding).

str1_3 (double) Alternative strategy of the first phenotype.

str2_2 (double) Strategy of the second phenotype using `H`, `Id`, `S`, `T`, `X`, `Y`, or `Z` gate (binary encoding).

str2_3 (double) Alternative strategy of the second phenotype.

str2_4 (double) Alternative strategy of the second phenotype.

str2_5 (double) Alternative strategy of the second phenotype.

str2_6 (double) Alternative strategy of the second phenotype.

str2_7 (double) Alternative strategy of the second phenotype.

batch_1 (double) Indicates whether predictions were made on full data or batch data.

TB_1 (double) Indicator for tuberculosis presence (1 = TB, 0 = no TB).

GR_1 (double) Indicator for genoresistance presence (1 = resistant, 0 = non-resistant).

Examples

```
data(qphen)
head(qphen)
```

`run_bb84_simulation` *BB84 Quantum Key Distribution (QKD) Simulation*

Description

Simulates the BB84 protocol for quantum key distribution (QKD), illustrating the difference in the Quantum Bit Error Rate (QBER) between an ideal channel (no interference) and a channel under eavesdropping (Eve). The simulation models the encoding, transmission, and measurement of quantum bits (qubits) following the original Bennett–Brassard (1984) protocol.

Usage

```
run_bb84_simulation(
  eavesdropping_active = FALSE,
  key_length = 1000,
  test_ratio = 0.2
)
```

Arguments

`eavesdropping_active`	Logical. If `TRUE`, the simulation includes an eavesdropper (Eve) who performs a measure-and-resend attack, introducing errors. If `FALSE`, the simulation runs under ideal, interference-free conditions. Default is `FALSE`.
`key_length`	Integer. Length of the quantum key sequence to be simulated. The default is 1000, which provides good statistical stability.
`test_ratio`	Numeric. Proportion (between 0 and 1) of bits that Alice and Bob compare to detect eavesdropping during the verification phase. Default is 0.2.

Details

The BB84 protocol proceeds through the following stages:

1. **Preparation:** Alice generates random bits and bases and encodes photons accordingly.
2. **Transmission:** Photons are sent over a quantum channel.
3. **Eavesdropping (optional):** Eve intercepts each photon, measures it using a random basis, and resends a new photon, introducing potential errors.
4. **Measurement:** Bob measures the incoming photons using his own random bases.
5. **Sifting:** Alice and Bob retain only bits measured with matching bases.
6. **Error estimation:** A random subset of the sifted bits is compared to estimate QBER.

If the measured QBER exceeds a security threshold (commonly 15–25%), it indicates that eavesdropping has occurred and the key must be discarded. In the absence of interference, the QBER should be close to zero.

Value

A list of class "BB84Simulation" containing:

QBER The observed Quantum Bit Error Rate between Alice's and Bob's bits.

Sifted_Length Number of bits retained after basis reconciliation.

Final_Key_Length Number of bits remaining after error testing.

Eavesdropping Logical flag indicating whether eavesdropping was active.

References

Bennett, C. H., & Brassard, G. (1984). *Quantum cryptography: Public key distribution and coin tossing.* Proceedings of IEEE International Conference on Computers, Systems and Signal Processing, 175–179.

Nielsen, M. A., & Chuang, I. L. (2010). *Quantum Computation and Quantum Information.* Cambridge University Press.

Rieffel, E. G., & Polak, W. H. (2011). *Quantum Computing: A Gentle Introduction.* MIT Press.

Examples

```
# Scenario 1: Perfect Channel (No Eavesdropping)
results_no_eve <- run_bb84_simulation(eavesdropping_active = FALSE)
cat("--- Scenario 1: No Eavesdropping ---\n")
cat(paste("Sifted Key Length:", results_no_eve$Sifted_Length, "\n"))
cat(paste("Final Key Length:", results_no_eve$Final_Key_Length, "\n"))
cat(paste("Quantum Bit Error Rate (QBER):", round(results_no_eve$QBER * 100, 2), "%\n"))
cat("Result: Secure key established successfully.\n")

# Scenario 2: Measure-and-Resend Attack
results_with_eve <- run_bb84_simulation(eavesdropping_active = TRUE)
cat("\n--- Scenario 2: With Eavesdropping ---\n")
cat(paste("Sifted Key Length:", results_with_eve$Sifted_Length, "\n"))
cat(paste("Final Key Length:", results_with_eve$Final_Key_Length, "\n"))
cat(paste("Quantum Bit Error Rate (QBER):", round(results_with_eve$QBER * 100, 2), "%\n"))

if (results_with_eve$QBER > 0.15) {
  cat("High QBER detected. Eavesdropping likely - key discarded.\n")
} else {
  cat("No eavesdropping detected (unlikely in theory).\n")
}
```

---

run_e91_simulation *E91 Quantum Key Distribution (QKD) Simulation*

Description

Simulates the E91 (Ekert 1991) quantum key distribution protocol using entangled particle pairs (EPR) and Bell's theorem (CHSH statistic) to ensure channel security. Security is established when the **Bell Inequality is violated**, i.e. when $|S| > 2$, indicating quantum behavior.

Usage

```
run_e91_simulation(
  eavesdropping_active = FALSE,
  key_length = 1000,
  noise_level = 0.5
)
```

Arguments

`eavesdropping_active`	Logical. If `TRUE`, the entanglement is partially destroyed, simulating an attack that forces the system toward the classical limit where $\|S\| \leq 2$. Default is `FALSE`.
`key_length`	Integer. Number of simulated EPR pairs (default = 1000).
`noise_level`	Numeric. Noise level (0-1) applied when `eavesdropping_active = TRUE`, reducing quantum correlation. A value of 0.5 represents partial decoherence. Default is 0.5.

Details

El E91 se diferencia de BB84 en que la detección del espía es **simultánea** a la generación de la clave. Los pares de bases son separados en dos conjuntos:

1. **Clave Secreta:** Pares con perfecta anti-correlación (ej. a2-b1, a3-b2).
2. **Test de Bell:** Pares restantes usados para calcular la esperanza $E(a_i, b_j)$ y el valor S.

Si S cae por debajo de 2 (límite clásico de Bell), se concluye que el entrelazamiento ha sido roto por Eve, y la clave debe descartarse.

Value

A list of class `"E91Simulation"` containing:

`S_Calculated` Observed Bell CHSH statistic.

`S_Theoretical` Quantum theoretical value ($\approx$ -2.8284).

`Bell_Violation` `TRUE` if $|S| > 2$ (secure), `FALSE` if $|S| \leq 2$ (insecure).

`Sifted_Key_Length` Number of bits retained for key formation.

`Eavesdropping` Indicates if an attack was simulated.

References

Ekert, A. K. (1991). *Quantum cryptography based on Bell's theorem.* Physical Review Letters, 67(6), 661.

Examples

```
# Escenario 1: Canal Cuántico Seguro (No Eavesdropping)
results_secure <- run_e91_simulation(eavesdropping_active = FALSE)
cat("--- Escenario 1: Sin Eavesdropping ---\n")
cat(paste("S Calculado:", round(results_secure$S_Calculated, 4), "\n"))
cat(paste("Violación de Bell:", results_secure$Bell_Violation, "\n"))

# Escenario 2: Ataque que Rompe el Entrelazamiento
results_attack <- run_e91_simulation(eavesdropping_active = TRUE)
cat("\n--- Escenario 2: Con Ataque ---\n")
cat(paste("S Calculado:", round(results_attack$S_Calculated, 4), "\n"))
cat(paste("Violación de Bell:", results_attack$Bell_Violation, "\n"))
```

`summary` *Summarize an InteractionClassification object*

Description

Computes summary statistics by classification group from an object of class `InteractionClassification`, including mean differences in viral load and CD4 counts, and the number of observations per cluster.

This function summarizes the payoffs object by classification.

Usage

```
summary(object, ...)

summary(object, ...)
```

Arguments

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

Value

A `data.frame` with one row per interaction cluster and the following columns:

classification Cluster label (as factor).

cds_diff_mean Mean of CD4 differences in the group.

vlogs_diff_mean Mean of viral load differences in the group.

n Number of observations in the group.

Examples

```
set.seed(42)
data(cd_3)
cd_data <- cd_3[,-1]
cd_result <- cds_diff(cd_data)
data(vl_3)
vl_data <- vl_3[,-1]
vl_result <- vlogs_diff(vl_data)
result <- InteractionClassification(cd_result = cd_result, vl_result = vl_result)
summary(result)
set.seed(42)
data(cd_3)
cd_data <- cd_3[,-1]
cd_result <- cds_diff(cd_data)
data(vl_3)
vl_data <- vl_3[,-1]
vl_result <- vlogs_diff(vl_data)
result <- InteractionClassification(cd_result = cd_result, vl_result = vl_result)
data(preds)
payoffs_results <- estimate_payoffs(result, preds)
summary(payoffs_results)
```

---

summary.InteractionClassification

Summarize an InteractionClassification object

Description

Computes summary statistics by classification group from an object of class `InteractionClassification`, including mean differences in viral load and CD4 counts, and the number of observations per cluster.

Usage

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

Arguments

`object`	An object of class `InteractionClassification` returned by the `InteractionClassification()` function.
`...`	Additional arguments passed to other methods (currently not used).

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).

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)
```

---

vl_diff *Create Mean Differences from Longitudinal Viral Load Data*

Description

This function calculates the mean differences of viral loads across time for each individual in the dataset.

Usage

```
vl_diff(vl_data)
```

Arguments

vl_data	A data frame of longitudinal viral load values per individual, where rows represent patients and columns represent sequential measurements across time (e.g., years or visits).

Value

An object of class "Interaction" with the following components:

vl3_diff Mean differences of raw viral load values.

Examples

```
data(vl_3)
vl_data <- vl_3[,-1]
result <- vl_diff(vl_data)
```

---

vlog_diff *Create Mean Differences from Logarithmic Viral Load Data*

Description

This function calculates the mean differences of logarithmic viral loads across time for each individual in the dataset.

Usage

```
vlog_diff(vl_data)
```

Arguments

vl_data	A data frame of longitudinal viral load values per individual, where rows represent patients and columns represent sequential measurements across time (e.g., years or visits).

Value

An object of class "Interaction" with the following components:

vlog3_diff Mean differences of logarithmic viral load values.

Examples

```
data(vl_3)
vl_data <- vl_3[,-1]
result <- vlog_diff(vl_data)
```

vlogs_diff *Create Mean Standardized Differences from Logarithmic Viral Load Data*

Description

This function calculates the mean standardized differences of logarithmic viral loads across time for each individual in the dataset.

Usage

```
vlogs_diff(vl_data)
```

Arguments

vl_data	A data frame of longitudinal viral load values per individual, where rows represent patients and columns represent sequential measurements across time (e.g., years or visits).

Value

An object of class "Interaction" with the following components:

vlogs3_diff Mean standardized differences of logarithmic viral load values.

Examples

```
data(vl_3)
vl_data <- vl_3[,-1]
result <- vlogs_diff(vl_data)
```

Index