Package ‘PRA’

January 20, 2025

Type Package

Title Project Risk Analysis

Version 0.3.0

Description Data analysis for Project Risk Management via the Second Moment Method, Monte Carlo Simulation, Contingency Analysis, Sensitivity Analysis, Earned Value Management, Learning Curves, Design Structure Matrices, and more.

Imports mc2d, minpack.lm, stats

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

URL `https://paulgovan.github.io/PRA/`, `https://github.com/paulgovan/PRA`

BugReports `https://github.com/paulgovan/PRA/issues`

Suggests ggplot2, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Paul Govan [aut, cre, cph] (<`https://orcid.org/0000-0002-1821-8492`>)

Maintainer Paul Govan <`paul.govan2@gmail.com`>

Repository CRAN

Date/Publication 2024-08-20 17:40:02 UTC

Contents

ac *Actual Cost (AC).*

Description

Actual Cost (AC).

Usage

```
ac(actual_costs, time_period)
```

Arguments

`actual_costs`	Vector of actual costs incurred at each time period.
`time_period`	Current time period.

Value

The function returns the Actual Cost (AC) of work completed.

Examples

```
# Set the actual costs and current time period for a toy project.
actual_costs <- c(9000, 18000, 36000, 70000, 100000)
time_period <- 3

# Calculate the AC and print the results.
ac <- ac(actual_costs, time_period)
cat("Actual Cost (AC):", ac, "\n")
```

`contingency` *Contingency Calculation.*

Description

Contingency Calculation.

Usage

```
contingency(sims, phigh = 0.95, pbase = 0.5)
```

Arguments

`sims`	List of results from a Monte Carlo simulation.
`phigh`	Percentile level for contingency calculation. Default is 0.95.
`pbase`	Base level for contingency calculation. Default is 0.5

Value

The function returns the value of calculated contingency.

Examples

```
# Set the number os simulations and the task distributions for a toy project.
num_sims <- 10000
task_dists <- list(
  list(type = "normal", mean = 10, sd = 2),  # Task A: Normal distribution
  list(type = "triangular", a = 5, b = 10, c = 15),  # Task B: Triangular distribution
  list(type = "uniform", min = 8, max = 12)  # Task C: Uniform distribution
)

# Set the correlation matrix for the correlations between tasks.
cor_mat <- matrix(c(
  1, 0.5, 0.3,
  0.5, 1, 0.4,
  0.3, 0.4, 1
), nrow = 3, byrow = TRUE)

# Run the Monte Carlo simulation.
results <- mcs(num_sims, task_dists, cor_mat)

# Calculate the contingency and print the results.
contingency <- contingency(results, phigh = 0.95, pbase = 0.50)
cat("Contingency based on 95th percentile and 50th percentile:", contingency)
```

`cor_matrix` *Generate Correlation Matrix.*

Description

Generate Correlation Matrix.

Usage

```
cor_matrix(num_samples = 100, num_vars = 5, dists = dists)
```

Arguments

`num_samples`	The number of samples to generate.
`num_vars`	The number of distributions to sample.
`dists`	A list describing each distribution.

Value

The function returns the correlation matrix for the distributions.

Examples

```
# List of probability distributions
dists <- list(
  normal = function(n) rnorm(n, mean = 0, sd = 1),
  uniform = function(n) runif(n, min = 0, max = 1),
  exponential = function(n) rexp(n, rate = 1),
  poisson = function(n) rpois(n, lambda = 1),
  binomial = function(n) rbinom(n, size = 10, prob = 0.5)
)

# Generate correlation matrix
cor_matrix <- cor_matrix(num_samples = 100, num_vars = 5, dists = dists)

# Print correlation matrix
print(cor_matrix)
```

`cpi` *Cost Performance Index (CPI).*

Description

Cost Performance Index (CPI).

Usage

```
cpi(ev, ac)
```

Arguments

`ev`	Earned Value.
`ac`	Actual Cost.

Value

The function returns the Cost Performance Index (CPI) of work completed.

Examples

```
# Set the BAC and actual % complete for an example project.
bac <- 100000
actual_per_complete <- 0.35

# Calcualte the EV
ev <- ev(bac, actual_per_complete)

# Set the actual costs and current time period and calculate the AC.
actual_costs <- c(9000, 18000, 36000, 70000, 100000)
time_period <- 3
ac <- ac(actual_costs, time_period)

# Calculate the CPI and print the results.
cpi <- cpi(ev, ac)
cat("Cost Performance Index (CPI):", cpi, "\n")
```

`cv` *Cost Variance (CV).*

Description

Cost Variance (CV).

Usage

```
cv(ev, ac)
```

Arguments

`ev`	Earned Value.
`ac`	Actual Cost.

Value

The function returns the Cost Variance (CV) of work completed.

Examples

```
# Set the BAC and actual % complete for an example project.
bac <- 100000
actual_per_complete <- 0.35

# Calcualte the EV
ev <- ev(bac, actual_per_complete)

# Set the actual costs and current time period and calculate the AC.
actual_costs <- c(9000, 18000, 36000, 70000, 100000)
time_period <- 3
ac <- ac(actual_costs, time_period)

# Calculate the CV and print the results.
cv <- cv(ev, ac)
cat("Cost Variance (CV):", cv, "\n")
```

---

ev *Earned Value (EV).*

Description

Earned Value (EV).

Usage

```
ev(bac, actual_per_complete)
```

Arguments

bac	Budget at Completion (BAC) (total planned budget).
actual_per_complete	Actual work completion percentage.

Value

The function returns the Earned Value (EV) of work completed.

Examples

```
# Set the BAC and actual % complete for a toy project.
bac <- 100000
actual_per_complete <- 0.35

# Calculate the EV and print the results.
ev <- ev(bac, actual_per_complete)
cat("Earned Value (EV):", ev, "\n")
```

`fit_sigmoidal` *Fit a Sigmoidal Model.*

Description

Fit a Sigmoidal Model.

Usage

```
fit_sigmoidal(data, x_col, y_col, model_type)
```

Arguments

`data`	A data frame containing the time (x_col) and completion (y_col) vectors.
`x_col`	The name of the time vector.
`y_col`	The name of the completion vector.
`model_type`	The name of the sigmoidal model (Pearl, Gompertz, or Logistic).

Value

The function returns a list of results for the sigmoidal model.

Examples

```
# Set up a data frame of time and completion percentage data
data <- data.frame(time = 1:10, completion = c(5, 15, 40, 60, 70, 75, 80, 85, 90, 95))

# Fit a logistic model to the data.
fit <- fit_sigmoidal(data, "time", "completion", "logistic")

# Use the model to predict future completion times.
predictions <- predict_sigmoidal(fit, seq(min(data$time), max(data$time),
  length.out = 100), "logistic")

# Plot the results.
p <- ggplot2::ggplot(data, ggplot2::aes_string(x = "time", y = "completion")) +
  ggplot2::geom_point() +
  ggplot2::geom_line(data = predictions, ggplot2::aes(x = x, y = pred), color = "red") +
  ggplot2::labs(title = "Fitted Logistic Model", x = "time", y = "completion %") +
  ggplot2::theme_minimal()
p
```

grandparent_dsm *Risk-based 'Grandparent' Design Structure Matrix (DSM).*

Description

Risk-based 'Grandparent' Design Structure Matrix (DSM).

Usage

```
grandparent_dsm(S, R)
```

Arguments

S	Resource-Task Matrix 'S' giving the links (arcs) between resources and tasks.
R	Risk-Resource Matrix 'R' giving the links (arcs) between risks and resources.

Value

The function returns the Risk-based 'Grandparent' DSM 'G' giving the number of risks shared between each task.

Examples

```
# Set the S and R matrices and print the results.
S <- matrix(c(1, 1, 0, 0, 1, 0, 0, 1, 1), nrow = 3, ncol = 3)
R <- matrix(c(1, 1, 1, 1, 0, 0), nrow = 2, ncol = 3)
cat("Resource-Task Matrix:\n")
print(S)
cat("\nRisk-Resource Matrix:\n")
print(R)
# Calculate the Risk-based Grandparent Matrix and print the results.
risk_dsm <- grandparent_dsm(S, R)
cat("\nRisk-based 'Grandparent' DSM:\n")
print(risk_dsm)
```

mcs *Monte Carlo Simulation.*

Description

Monte Carlo Simulation.

Usage

```
mcs(num_sims, task_dists, cor_mat = NULL)
```

Arguments

`num_sims`	The number of simulations.
`task_dists`	A list of lists describing each task distribution.
`cor_mat`	The correlation matrix for the tasks (Optional).

Value

The function returns a list of the total mean, variance, standard deviation, and percentiles for the project.

Examples

```
# Set the number of simulations and task distributions for a toy project.
num_sims <- 10000
task_dists <- list(
  list(type = "normal", mean = 10, sd = 2),  # Task A: Normal distribution
  list(type = "triangular", a = 5, b = 10, c = 15),  # Task B: Triangular distribution
  list(type = "uniform", min = 8, max = 12)  # Task C: Uniform distribution
)

# Set the correlation matrix for the correlations between tasks.
cor_mat <- matrix(c(
  1, 0.5, 0.3,
  0.5, 1, 0.4,
  0.3, 0.4, 1
), nrow = 3, byrow = TRUE)

# Run the Monte Carlo sumulation and print the results.
results <- mcs(num_sims, task_dists, cor_mat)
cat("Mean Total Duration:", results$total_mean, "\n")
cat("Variance of Total Variance:", results$total_variance, "\n")
cat("Standard Deviation of Total Duration:", results$total_sd, "\n")
cat("5th Percentile:", results$percentiles[1], "\n")
cat("Median (50th Percentile):", results$percentiles[2], "\n")
cat("95th Percentile:", results$percentiles[3], "\n")
hist(results$total_distribution, breaks = 50, main = "Distribution of Total Project Duration",
  xlab = "Total Duration", col = "skyblue", border = "white")
```

`parent_dsm` *Resource-based 'Parent' Design Structure Matrix (DSM).*

Description

Resource-based 'Parent' Design Structure Matrix (DSM).

Usage

```
parent_dsm(S)
```

Arguments

`S`	Resource-Task Matrix 'S' giving the links (arcs) between resources and tasks.

Value

The function returns the Resource-based 'Parent' DSM 'P' giving the number of resources shared between each task.

Examples

```
# Set the S matrix for a toy project and print the results.
s <- matrix(c(1, 1, 0, 0, 1, 0, 0, 1, 1), nrow = 3, ncol = 3)
cat("Resource-Task Matrix:\n")
print(s)

# Calculate the Resource-based Parent DSM and print the results.
resource_dsm <- parent_dsm(s)
cat("\nResource-based 'Parent' DSM:\n")
print(resource_dsm)
```

`predict_sigmoidal` *Predict a Sigmoidal Function.*

Description

Predict a Sigmoidal Function.

Usage

```
predict_sigmoidal(fit, x_range, model_type)
```

Arguments

`fit`	A list containing the results of a sigmoidal model.
`x_range`	A vector of time values for the prediction.
`model_type`	The type of model (Pearl, Gompertz, or Logistic) for the prediction.

Value

The function returns a table of results containing the time and predicted values.

Examples

```
# Set up a data frame of time and completion percentage data
data <- data.frame(time = 1:10, completion = c(5, 15, 40, 60, 70, 75, 80, 85, 90, 95))

# Fit a logistic model to the data.
fit <- fit_sigmoidal(data, "time", "completion", "logistic")

# Use the model to predict future completion times.
predictions <- predict_sigmoidal(fit, seq(min(data$time), max(data$time),
  length.out = 100), "logistic")

# Plot the results.
p <- ggplot2::ggplot(data, ggplot2::aes_string(x = "time", y = "completion")) +
  ggplot2::geom_point() +
  ggplot2::geom_line(data = predictions, ggplot2::aes(x = x, y = pred), color = "red") +
  ggplot2::labs(title = "Fitted Logistic Model", x = "time", y = "completion %") +
  ggplot2::theme_minimal()
p
```

`pv` *Planned Value (PV).*

Description

Planned Value (PV).

Usage

```
pv(bac, schedule, time_period)
```

Arguments

`bac`	Budget at Completion (BAC) (total planned budget).
`schedule`	Vector of planned work completion (in terms of percentage) at each time period.
`time_period`	Current time period.

Value

The function returns the Planned Value (PV) of work completed.

Examples

```
# Set the BAC, schedule, and current time period for a toy project.
bac <- 100000
schedule <- c(0.1, 0.2, 0.4, 0.7, 1.0)
time_period <- 3

# Calculate the PV and print the results.
pv <- pv(bac, schedule, time_period)
cat("Planned Value (PV):", pv, "\n")
```

`sensitivity` *Sensitivity Analysis.*

Description

Sensitivity Analysis.

Usage

```
sensitivity(task_dists, cor_mat = NULL)
```

Arguments

`task_dists`	A list of lists describing each task distribution.
`cor_mat`	The correlation matrix for the tasks (Optional).

Value

The function returns a vector of sensitivity results with respect to each task.

Examples

```
# Set the task distributions for a toy project.
task_dists <- list(
  list(type = "normal", mean = 10, sd = 2),  # Task A: Normal distribution
  list(type = "triangular", a = 5, b = 15, c = 10),  # Task B: Triangular distribution
  list(type = "uniform", min = 8, max = 12)  # Task C: Uniform distribution
)

# Set the correlation matrix between the tasks.
cor_mat <- matrix(c(
  1, 0.5, 0.3,
 0.5, 1, 0.4,
  0.3, 0.4, 1
), nrow = 3, byrow = TRUE)

# Calculate the sensitivity of each task and print the results
sensitivity_results <- sensitivity(task_dists, cor_mat)
cat("Sensitivity of the variance in total cost with respect to the variance in each task cost:\n")
print(sensitivity_results)

# Build a vertical barchart and display the results.
data <- data.frame(
  Tasks = c('A', 'B', 'C'),
  Sensitivity = sensitivity_results
)
barplot(height=data$Sensitivity, names=data$Tasks, col='skyblue',
        horiz=TRUE, xlab = 'Sensitivity', ylab = 'Tasks')
```

smm *Second Moment Analysis.*

Description

Second Moment Analysis.

Usage

```
smm(mean, var, cor_mat = NULL)
```

Arguments

`mean`	The mean vector.
`var`	The variance vector.
`cor_mat`	The correlation matrix (optional).

Value

The function returns a list of the total mean, variance, and standard deviation for the project.

Examples

```
# Set the mean vector, variance vector, and correlation matrix for a toy project.
mean <- c(10, 15, 20)
var <- c(4, 9, 16)
cor_mat <- matrix(c(
  1, 0.5, 0.3,
  0.5, 1, 0.4,
  0.3, 0.4, 1
), nrow = 3, byrow = TRUE)

# Use the Second Moment Method to estimate the results for the project.
result <- smm(mean, var, cor_mat)
print(result)
```

spi *Schedule Performance Index (SPI).*

Description

Schedule Performance Index (SPI).

Usage

```
spi(ev, pv)
```

Arguments

`ev`	Earned Value.
`pv`	Planned Value.

Value

The function returns the Schedule Performance Index (SPI) of work completed.

Examples

```
# Set the BAC, schedule, and current time period for an example project.
bac <- 100000
schedule <- c(0.1, 0.2, 0.4, 0.7, 1.0)
time_period <- 3

# Calculate the PV.
pv <- pv(bac, schedule, time_period)

# Set the actual % complete and calculate the EV.
actual_per_complete <- 0.35
ev <- ev(bac, actual_per_complete)

# Calculate the SPI and print the results.
spi <- spi(ev, pv)
cat("Schedule Performance Index (SPI):", spi, "\n")
```

`sv` *Schedule Variance (SV).*

Description

Schedule Variance (SV).

Usage

```
sv(ev, pv)
```

Arguments

`ev`	Earned Value.
`pv`	Planned Value.

Value

The function returns the Schedule Variance (SV) of work completed.

Examples

```
# Set the BAC, schedule, and current time period for an example project.
bac <- 100000
schedule <- c(0.1, 0.2, 0.4, 0.7, 1.0)
time_period <- 3

# Calculate the PV.
pv <- pv(bac, schedule, time_period)

# Set the actual % complete and calculate the EV.
actual_per_complete <- 0.35
ev <- ev(bac, actual_per_complete)

# Calculate the SV and print the results.
sv <- sv(ev, pv)
cat("Schedule Variance (SV):", sv, "\n")
```

Index