SWWC_CQAWWC_CQAWP

Xiande Yang, Wendy Lou, and Mohamed Chan

1 R Package and Data Loading

All functions outputs are in data frame format of one row record. If users want to display bootstrap output, you need to rewrite the output in list format.

```
package_vector <- c("quarto", "rmarkdown", "shiny", "rlang", "tidyr", "DBI", "odbc", "open
                    "stringr", "ggplot2", "forcats", "readxl", "formattable", "easyanova",
                    "kableExtra", "lme4", "AER", "dunn.test", "boot", "lmtest", "gridExtra
                    "webshot2", "chromote",  "DiagrammeR", "DiagrammeRsvg", "rsvg", "rlang

# "CleaningValidation", we do not use this package for this manuscript

check_and_install <- function(pkg) {
  if (!require(pkg, character.only = TRUE)) {
    install.packages(pkg, dependencies = TRUE, repos = "https://cran.r-project.org")
    require(pkg, character.only = TRUE)
  }
}
lapply(package_vector, require, character.only=T)
Eq_DAR <- read_excel("C:\\Users\\xyang1\\AAPSPharmSciTech\\USL_unification-main\\Equipment
Eq_CAR <- read_excel("C:\\Users\\xyang1\\AAPSPharmSciTech\\USL_unification-main\\Equipment
Eq_Mic <- read_excel("C:\\Users\\xyang1\\AAPSPharmSciTech\\USL_unification-main\\Equipment
Eq_DAR_20 <- read_excel("C:\\Users\\xyang1\\AAPSPharmSciTech\\USL_unification-main\\Equipm
Eq_DAR_169 <- read_excel("C:\\Users\\xyang1\\AAPSPharmSciTech\\USL_unification-main\\Equip
Eq_DAR_122 <- read_excel("C:\\Users\\xyang1\\AAPSPharmSciTech\\USL_unification-main\\Equip
```

2.1 Ppu for one data set with USL-normalization

```
Ppu_KDEDPonUSLND <- function(data, Residue, USL, BW = "Silver1.06") {
  library(dplyr)
  library(rlang)
  residue_quo <- enquo(Residue)
  usl_quo <- enquo(USL)
  data_clean <- data %>%
    filter(!is.na(!!residue_quo), !is.na(!!usl_quo)) %>%
    mutate(Residue_Pct = (!!residue_quo / !!usl_quo) * 100) # USL-normalization step

  if (nrow(data_clean) == 0) stop("No valid rows remaining after removing NAs in Residue o
   x <- data_clean$Residue_Pct
  n <- length(x)
  s <- sd(x)
  if (is.character(BW)) {
    BW <- match.arg(BW, choices = c("Silver1.06", "Silver0.9", "Silver0.9IQR"))
    h <- switch(BW,
                "Silver1.06" = 1.06 * s / n^(1/5),
                "Silver0.9" = 0.9 * s / n^(1/5),
                "Silver0.9IQR" = {
                  iqr_val <- IQR(x)
                  sigma <- min(s, iqr_val / 1.34)
                  0.9 * sigma / n^(1/5)
                })
    bw_method <- BW
  } else if (is.numeric(BW) && length(BW) == 1 && BW > 0) {
    h <- BW
    bw_method <- "User-defined"
  } else {
    stop("BW must be numeric > 0 or one of: 'Silver1.06', 'Silver0.9', 'Silver0.9IQR'.")
  }
  kde <- density(x, bw = h, n = 2^20) # the default kernel is Gaussian
  fx <- kde$y
  x_vals <- kde$x
  dx <- diff(x_vals)[1]
  Fx <- cumsum(fx) * dx
  Fx <- Fx / max(Fx)
  inv_cdf <- approxfun(Fx, x_vals, rule = 2) # we did not use uniroot() which maybe better
  P0.5 <- inv_cdf(0.5)
  P0.99865 <- inv_cdf(0.99865)
```

```
  Ppu <- (100 - P0.5) / (P0.99865 - P0.5)
  df_result <- data.frame(
    Ppu = round(Ppu, 3),
    P0.5 = round(P0.5, 3),
    P0.99865 = round(P0.99865, 3),
    N = n,
    Sample_SD = round(s, 3),
    Bandwidth_Method = bw_method,
    Bandwidth_Value = round(h, 5)
  )
  return(df_result)
}
```

2.2 Ppu and confidence interval with bootstrap for one data set with USL-normalization

```
Ppu_BAKDEDPonUSLND <- function(data, Residue, USL, BW = "Silver1.06", n_boot = 1000, conf_
  library(rlang)
  Residue_enquo <- enquo(Residue)
  USL_enquo <- enquo(USL)
  set.seed(seed)
  ppu_full <- Ppu_KDEDPonUSLND(data, !!Residue_enquo, !!USL_enquo, BW)$Ppu
  n <- nrow(data)
  ppu_boot <- numeric(n_boot)
  for (i in seq_len(n_boot)) {
    data_boot <- data[sample(seq_len(n), size = n, replace = TRUE), ]
    ppu_boot[i] <- tryCatch({
      Ppu_KDEDPonUSLND(data_boot, !!Residue_enquo, !!USL_enquo, BW)$Ppu
    }, error = function(e) NA_real_)
  }
  ppu_boot <- ppu_boot[!is.na(ppu_boot)]
  alpha <- (1 - conf_level)
  cil <- quantile(ppu_boot, probs = alpha / 2, names = FALSE)
  ciu <- quantile(ppu_boot, probs = 1 - alpha / 2, names = FALSE)
  result_df <- data.frame(
    Ppu = ppu_full,
    CI_lower = cil,
    CI_upper = ciu,
    conf_level = conf_level,
```

```
    n_boot = length(ppu_boot),
    stringsAsFactors = FALSE
  )
  return(result_df)
}
```

3.1 Ppu CQAWWC for one to three data with USL-normalization

```
Ppu_CQAWWC_KDEDPonUSLND <- function(data1, Residue1, USL1,
                                    data2 = NULL, Residue2 = NULL, USL2 = NULL,
                                    data3 = NULL, Residue3 = NULL, USL3 = NULL,
                                    BW = "Silver1.06") {
  ppu_named <- c()
  ppu1 <- Ppu_KDEDPonUSLND(data1, {{ Residue1 }}, {{ USL1 }}, BW)$Ppu
  ppu_named <- c(ppu_named, Ppu_Data1 = ppu1)

  if (!is.null(data2) && !rlang::quo_is_null(rlang::enquo(Residue2)) && !rlang::quo_is_nul
    ppu2 <- Ppu_KDEDPonUSLND(data2, {{ Residue2 }}, {{ USL2 }}, BW)$Ppu
    ppu_named <- c(ppu_named, Ppu_Data2 = ppu2)
  }

  if (!is.null(data3) && !rlang::quo_is_null(rlang::enquo(Residue3)) && !rlang::quo_is_nul
    ppu3 <- Ppu_KDEDPonUSLND(data3, {{ Residue3 }}, {{ USL3 }}, BW)$Ppu
    ppu_named <- c(ppu_named, Ppu_Data3 = ppu3)
  }

  min_ppu <- min(ppu_named, na.rm = TRUE)

  output <- as.data.frame(as.list(c(Ppu = min_ppu, ppu_named)))

  return(output)
}
```

3.2 Ppu CQAWWC with confidence interval for one to three data with bootstrap and USL-normalization

```
Ppu_CQAWWC_BAKDEDPonUSLND <- function(data1, Residue1, USL1,
                                      data2 = NULL, Residue2 = NULL, USL2 = NULL,
                                      data3 = NULL, Residue3 = NULL, USL3 = NULL,
                                      BW = "Silver1.06",
                                      n_boot = 1000,
                                      conf_level = 0.95,
                                      seed = NULL) {
  library(dplyr)
  library(rlang)

  if (!is.null(seed)) {
    set.seed(seed)
  }

  clean_data <- function(data, residue, usl) {
    residue_quo <- enquo(residue)
    usl_quo <- enquo(usl)

    data %>%
      filter(!is.na(!!residue_quo), !is.na(!!usl_quo))
  }

  data1_clean <- clean_data(data1, {{Residue1}}, {{USL1}})
  data2_clean <- if (!is.null(data2)) clean_data(data2, {{Residue2}}, {{USL2}}) else NULL
  data3_clean <- if (!is.null(data3)) clean_data(data3, {{Residue3}}, {{USL3}}) else NULL

  point_estimate_all <- Ppu_CQAWWC_KDEDPonUSLND(data1_clean, {{Residue1}}, {{USL1}},
                                                data2_clean, {{Residue2}}, {{USL2}},
                                                data3_clean, {{Residue3}}, {{USL3}},
                                                BW = BW)
  point_estimate <- point_estimate_all$Ppu


  bootstrap_ppus <- replicate(n_boot, {
    boot_data1 <- data1_clean %>% sample_n(size = nrow(data1_clean), replace = TRUE)
    boot_data2 <- if (!is.null(data2_clean)) data2_clean %>% sample_n(size = nrow(data2_cl
    boot_data3 <- if (!is.null(data3_clean)) data3_clean %>% sample_n(size = nrow(data3_cl
```

```
    tryCatch({
      res <- Ppu_CQAWWC_KDEDPonUSLND(boot_data1, {{Residue1}}, {{USL1}},
                                     boot_data2, {{Residue2}}, {{USL2}},
                                     boot_data3, {{Residue3}}, {{USL3}},
                                     BW = BW)
      res$Ppu
    }, error = function(e) NA_real_)
  })

  bootstrap_ppus <- na.omit(bootstrap_ppus)

  alpha <- 1 - conf_level
  ci_lower <- quantile(bootstrap_ppus, probs = alpha / 2, names = FALSE)
  ci_upper <- quantile(bootstrap_ppus, probs = 1 - alpha / 2, names = FALSE)

  df_result <- data.frame(
    Ppu = round(point_estimate, 3),
    CI_lower = round(ci_lower, 3),
    n_boot = n_boot
  )

  return(df_result)
}
```

3.3 Ppu CQAWWC monitoring model with training Ppu, monitoring thresholod CIL, and monitoring output Ppu

```
CQAWWC_BAKDEDPonUSLND_CVStage3Monitoring <- function(
    data1, Residue1, USL1, Fiscal_Year1,
    data2 = NULL, Residue2 = NULL, USL2 = NULL, Fiscal_Year2 = NULL,
    data3 = NULL, Residue3 = NULL, USL3 = NULL, Fiscal_Year3 = NULL,
    Time_cut,
    CIL = NULL,
    BW = "Silver1.06",
    initial_boot = 1000,
    max_boot = 10000,
    conf_level = 0.95,
    seed = NULL
) {
```

```r
library(dplyr)

if (!is.null(seed)) set.seed(seed)

clean_na_residue <- function(data, residue_col) {
  data %>% filter(!is.na({{ residue_col }}))
}

split_by_fiscal_year <- function(data, fiscal_year_col, time_cut) {
  training <- data %>%
    filter(!is.na({{ fiscal_year_col }})) %>%
    filter({{ fiscal_year_col }} < time_cut)

  testing <- data %>%
    filter(!is.na({{ fiscal_year_col }})) %>%
    filter({{ fiscal_year_col }} >= time_cut)

  list(training = training, testing = testing)
}

split1 <- split_by_fiscal_year(data1, {{ Fiscal_Year1 }}, Time_cut)
TrainingData1 <- clean_na_residue(split1$training, {{ Residue1 }})
TestingData1 <- clean_na_residue(split1$testing, {{ Residue1 }})

if (!is.null(data2)) {
  split2 <- split_by_fiscal_year(data2, {{ Fiscal_Year2 }}, Time_cut)
  TrainingData2 <- clean_na_residue(split2$training, {{ Residue2 }})
  TestingData2 <- clean_na_residue(split2$testing, {{ Residue2 }})
} else {
  TrainingData2 <- NULL
  TestingData2 <- NULL
}

if (!is.null(data3)) {
  split3 <- split_by_fiscal_year(data3, {{ Fiscal_Year3 }}, Time_cut)
  TrainingData3 <- clean_na_residue(split3$training, {{ Residue3 }})
  TestingData3 <- clean_na_residue(split3$testing, {{ Residue3 }})
} else {
  TrainingData3 <- NULL
  TestingData3 <- NULL
}
```

```
if (is.null(CIL) || CIL < 1) {
  ppu_ci_obj <- Ppu_CQAWWC_BAKDEDPonUSLND(
    data1 = TrainingData1, Residue1 = {{ Residue1 }}, USL1 = {{ USL1 }},
    data2 = TrainingData2, Residue2 = {{ Residue2 }}, USL2 = {{ USL2 }},
    data3 = TrainingData3, Residue3 = {{ Residue3 }}, USL3 = {{ USL3 }},
    BW = BW,
    n_boot = initial_boot,
    conf_level = conf_level,
    seed = seed
  )

  if (ppu_ci_obj$CI_lower < 1) {
    message("Initial CI lower bound < 1, increasing bootstrap to ", max_boot)
    ppu_ci_obj <- Ppu_CQAWWC_BAKDEDPonUSLND(
      data1 = TrainingData1, Residue1 = {{ Residue1 }}, USL1 = {{ USL1 }},
      data2 = TrainingData2, Residue2 = {{ Residue2 }}, USL2 = {{ USL2 }},
      data3 = TrainingData3, Residue3 = {{ Residue3 }}, USL3 = {{ USL3 }},
      BW = BW,
      n_boot = max_boot,
      conf_level = conf_level,
      seed = seed
    )
    if (ppu_ci_obj$CI_lower < 1) {
      warning("CIL still < 1 after ", max_boot, " bootstraps. Data quality or size may b
    }
  }
  CIL <- ppu_ci_obj$CI_lower
} else {
  training_ppu <- Ppu_CQAWWC_KDEDPonUSLND(
    data1 = TrainingData1, Residue1 = {{ Residue1 }}, USL1 = {{ USL1 }},
    data2 = TrainingData2, Residue2 = {{ Residue2 }}, USL2 = {{ USL2 }},
    data3 = TrainingData3, Residue3 = {{ Residue3 }}, USL3 = {{ USL3 }},
    BW = BW
  )$Ppu

  ppu_ci_obj <- list(
    Ppu = training_ppu,
    CI_lower = CIL

  )
}
```

```
  CombinedData1 <- bind_rows(TrainingData1, TestingData1)
  CombinedData2 <- if (!is.null(TrainingData2)) bind_rows(TrainingData2, TestingData2) els
  CombinedData3 <- if (!is.null(TrainingData3)) bind_rows(TrainingData3, TestingData3) els

  testing_ppu_result <- Ppu_CQAWWC_KDEDPonUSLND(
    data1 = CombinedData1, Residue1 = {{ Residue1 }}, USL1 = {{ USL1 }},
    data2 = CombinedData2, Residue2 = {{ Residue2 }}, USL2 = {{ USL2 }},
    data3 = CombinedData3, Residue3 = {{ Residue3 }}, USL3 = {{ USL3 }},
    BW = BW
  )

  Testing_Ppu <- testing_ppu_result$Ppu

  decision <- if (Testing_Ppu >= CIL) {
    "Cleaning process is capable."
  } else if (Testing_Ppu >= 1 && Testing_Ppu < CIL) {
    "Cleaning process is capable with low confidence - warning triggered."
  } else {
    "Cleaning process is NOT capable."
  }
  output <- data.frame(
    Ppu_training = ppu_ci_obj$Ppu,
    Ppu_threshold = CIL,
    Ppu_monitoring = Testing_Ppu,
    decision = decision

  )
  return(output)
}
```

4.1 Ppu CQAWP for one to three data with USL-normalization

```
Ppu_CQAWP_KDEDPonUSLND <- function(data1, Residue1, USL1,
                                   data2 = NULL, Residue2 = NULL, USL2 = NULL,
                                   data3 = NULL, Residue3 = NULL, USL3 = NULL,
                                   BW = "Silver1.06") {
  library(dplyr)
  library(rlang)
  clean_and_rename <- function(data, residue, usl) {
```

```
    residue_quo <- enquo(residue)
    usl_quo <- enquo(usl)
    data %>%
      filter(!is.na(!!residue_quo), !is.na(!!usl_quo)) %>%
      select(Residue = !!residue_quo, USL = !!usl_quo)
  }

  data1_clean <- clean_and_rename(data1, {{Residue1}}, {{USL1}})

  if (!is.null(data2) && !rlang::quo_is_null(rlang::enquo(Residue2)) && !rlang::quo_is_nul
    data2_clean <- clean_and_rename(data2, {{Residue2}}, {{USL2}})
  } else {
    data2_clean <- NULL
  }

  if (!is.null(data3) && !rlang::quo_is_null(rlang::enquo(Residue3)) && !rlang::quo_is_nul
    data3_clean <- clean_and_rename(data3, {{Residue3}}, {{USL3}})
  } else {
    data3_clean <- NULL
  }

  pooled_data <- data1_clean

  if (!is.null(data2_clean)) {
    pooled_data <- bind_rows(pooled_data, data2_clean)
  }

  if (!is.null(data3_clean)) {
    pooled_data <- bind_rows(pooled_data, data3_clean)
  }
  ppu_result <- Ppu_KDEDPonUSLND(
    data = pooled_data,
    Residue = Residue,
    USL = USL,
    BW = BW
  )

  return(ppu_result)
}
```

4.2 Ppu CQAWP for one to three data with bootstrap and USL-normalization

```
Ppu_CQAWP_BAKDEDPonUSLND <- function(data1, Residue1, USL1,
                                     data2 = NULL, Residue2 = NULL, USL2 = NULL,
                                     data3 = NULL, Residue3 = NULL, USL3 = NULL,
                                     BW = "Silver1.06",
                                     n_boot = 1000,
                                     conf_level = 0.95,
                                     seed = NULL) {
  library(dplyr)
  library(rlang)
  if (!is.null(seed)) set.seed(seed)
  point_estimate <- Ppu_CQAWP_KDEDPonUSLND(data1, {{Residue1}}, {{USL1}},
                                           data2, {{Residue2}}, {{USL2}},
                                           data3, {{Residue3}}, {{USL3}},
                                           BW = BW)$Ppu

  clean_data <- function(data, residue, usl) {
    residue_quo <- enquo(residue)
    usl_quo <- enquo(usl)
    data %>%
      filter(!is.na(!!residue_quo), !is.na(!!usl_quo))
  }
  data1_clean <- clean_data(data1, {{Residue1}}, {{USL1}})
  data2_clean <- if (!is.null(data2)) clean_data(data2, {{Residue2}}, {{USL2}}) else NULL
  data3_clean <- if (!is.null(data3)) clean_data(data3, {{Residue3}}, {{USL3}}) else NULL
  Residue1_quo <- enquo(Residue1)
  USL1_quo <- enquo(USL1)
  Residue2_quo <- if (!missing(Residue2)) enquo(Residue2) else NULL
  USL2_quo <- if (!missing(USL2)) enquo(USL2) else NULL
  Residue3_quo <- if (!missing(Residue3)) enquo(Residue3) else NULL
  USL3_quo <- if (!missing(USL3)) enquo(USL3) else NULL

  bootstrap_ppus <- replicate(n_boot, {
    boot_data1 <- data1_clean %>% sample_n(size = nrow(data1_clean), replace = TRUE)
    boot_data2 <- if (!is.null(data2_clean)) data2_clean %>% sample_n(size = nrow(data2_cl
    boot_data3 <- if (!is.null(data3_clean)) data3_clean %>% sample_n(size = nrow(data3_cl
    tryCatch({
      Ppu_CQAWP_KDEDPonUSLND(
        boot_data1, !!Residue1_quo, !!USL1_quo,
```

```
        boot_data2, !!Residue2_quo, !!USL2_quo,
        boot_data3, !!Residue3_quo, !!USL3_quo,
        BW = BW)$Ppu
    }, error = function(e) NA_real_)
  })
  bootstrap_ppus <- na.omit(bootstrap_ppus)
  alpha <- 1 - conf_level
  ci_lower <- quantile(bootstrap_ppus, probs = alpha / 2, names = FALSE)
  ci_upper <- quantile(bootstrap_ppus, probs = 1 - alpha / 2, names = FALSE)
    result_df <- data.frame(
    Ppu = round(point_estimate, 3),
    CI_lower = round(ci_lower, 3),
    n_boot = length(bootstrap_ppus),
    conf_level = conf_level,
    stringsAsFactors = FALSE
  )
   return(result_df)
}
```

4.3 Ppu CQAWP monitoring model with training, CIL, and monitoring

```
CQAWP_BAKDEDPonUSLND_CVStage3Monitoring <- function(
    data1, Residue1, USL1, Fiscal_Year1,
    data2 = NULL, Residue2 = NULL, USL2 = NULL, Fiscal_Year2 = NULL,
    data3 = NULL, Residue3 = NULL, USL3 = NULL, Fiscal_Year3 = NULL,
    Time_cut,
    CIL = NULL,
    BW = "Silver1.06",
    initial_boot = 1000,
    max_boot = 10000,
    conf_level = 0.95,
    seed = NULL
) {
  library(dplyr)

  if (!is.null(seed)) set.seed(seed)

  clean_na_residue <- function(data, residue_col) {
```

```
  data %>% filter(!is.na({{ residue_col }}))
}

split_by_fiscal_year <- function(data, fiscal_year_col, time_cut) {
  training <- data %>%
    filter(!is.na({{ fiscal_year_col }})) %>%
    filter({{ fiscal_year_col }} < time_cut)

  testing <- data %>%
    filter(!is.na({{ fiscal_year_col }})) %>%
    filter({{ fiscal_year_col }} >= time_cut)

  list(training = training, testing = testing)
}

split1 <- split_by_fiscal_year(data1, {{ Fiscal_Year1 }}, Time_cut)
TrainingData1 <- clean_na_residue(split1$training, {{ Residue1 }})
TestingData1 <- clean_na_residue(split1$testing, {{ Residue1 }})

if (!is.null(data2)) {
  split2 <- split_by_fiscal_year(data2, {{ Fiscal_Year2 }}, Time_cut)
  TrainingData2 <- clean_na_residue(split2$training, {{ Residue2 }})
  TestingData2 <- clean_na_residue(split2$testing, {{ Residue2 }})
} else {
  TrainingData2 <- NULL
  TestingData2 <- NULL
}

if (!is.null(data3)) {
  split3 <- split_by_fiscal_year(data3, {{ Fiscal_Year3 }}, Time_cut)
  TrainingData3 <- clean_na_residue(split3$training, {{ Residue3 }})
  TestingData3 <- clean_na_residue(split3$testing, {{ Residue3 }})
} else {
  TrainingData3 <- NULL
  TestingData3 <- NULL
}

if (is.null(CIL) || CIL < 1) {
  ppu_ci_obj <- Ppu_CQAWP_BAKDEDPonUSLND(
    data1 = TrainingData1, Residue1 = {{ Residue1 }}, USL1 = {{ USL1 }},
    data2 = TrainingData2, Residue2 = {{ Residue2 }}, USL2 = {{ USL2 }},
```

```
    data3 = TrainingData3, Residue3 = {{ Residue3 }}, USL3 = {{ USL3 }},
    BW = BW,
    n_boot = initial_boot,
    conf_level = conf_level,
    seed = seed
  )

  if (ppu_ci_obj$CI_lower < 1) {
    message("Initial CI lower bound < 1, increasing bootstrap to ", max_boot)
    ppu_ci_obj <- Ppu_CQAWP_BAKDEDPonUSLND(
      data1 = TrainingData1, Residue1 = {{ Residue1 }}, USL1 = {{ USL1 }},
      data2 = TrainingData2, Residue2 = {{ Residue2 }}, USL2 = {{ USL2 }},
      data3 = TrainingData3, Residue3 = {{ Residue3 }}, USL3 = {{ USL3 }},
      BW = BW,
      n_boot = max_boot,
      conf_level = conf_level,
      seed = seed
    )
    if (ppu_ci_obj$CI_lower < 1) {
      warning("CIL still < 1 after ", max_boot, " bootstraps. Data quality or size may b
    }
  }
  CIL <- ppu_ci_obj$CI_lower
} else {
  training_ppu <- Ppu_CQAWP_KDEDPonUSLND(
    data1 = TrainingData1, Residue1 = {{ Residue1 }}, USL1 = {{ USL1 }},
    data2 = TrainingData2, Residue2 = {{ Residue2 }}, USL2 = {{ USL2 }},
    data3 = TrainingData3, Residue3 = {{ Residue3 }}, USL3 = {{ USL3 }},
    BW = BW
  )$Ppu

  ppu_ci_obj <- list(
    Ppu = training_ppu,
    CI_lower = CIL
  )
}

CombinedData1 <- bind_rows(TrainingData1, TestingData1)
CombinedData2 <- if (!is.null(TrainingData2)) bind_rows(TrainingData2, TestingData2) els
CombinedData3 <- if (!is.null(TrainingData3)) bind_rows(TrainingData3, TestingData3) els
```

```
  testing_ppu_result <- Ppu_CQAWP_KDEDPonUSLND(
    data1 = CombinedData1, Residue1 = {{ Residue1 }}, USL1 = {{ USL1 }},
    data2 = CombinedData2, Residue2 = {{ Residue2 }}, USL2 = {{ USL2 }},
    data3 = CombinedData3, Residue3 = {{ Residue3 }}, USL3 = {{ USL3 }},
    BW = BW
  )

  Testing_Ppu <- testing_ppu_result$Ppu

  decision <- if (Testing_Ppu >= CIL) {
    "Cleaning process is capable."
  } else if (Testing_Ppu >= 1 && Testing_Ppu < CIL) {
    "Cleaning process is capable with low confidence - warning triggered."
  } else {
    "Cleaning process is NOT capable."
  }

  output <- data.frame(
    Ppu_training = ppu_ci_obj$Ppu,
    Ppu_threshold = CIL,
    Ppu_monitoring = Testing_Ppu,
    Performance_conclusion = decision,
    stringsAsFactors = FALSE
  )

  return(output)
}
```

5.1 Tradition Ppu by KDEDP (ISO2154-4 Method)

If you want to use this method, you have to split your data to subgroup data sets each has a fixed USL.

```
Ppu_KDEDP <- function(data, Residue, USL, BW = "Silver1.06") {
  library(dplyr)
  library(rlang)

  residue_quo <- enquo(Residue)
  usl_quo <- enquo(USL)
  data_clean <- data %>%
```

```
  filter(!is.na(!!residue_quo), !is.na(!!usl_quo)) %>%
  mutate(
    Residue_Val = !!residue_quo,
    USL_Val = !!usl_quo
  )

if (nrow(data_clean) == 0) stop("No valid rows remaining after removing NAs in Residue o
x <- data_clean$Residue_Val
usl_val <- data_clean$USL_Val[1]  # Assume constant USL
n <- length(x)
#s <- sqrt(mean((x - mean(x))^2))
s <- sd(x)
if (is.character(BW)) {
  BW <- match.arg(BW, choices = c("Silver1.06", "Silver0.9", "Silver0.9IQR"))
  h <- switch(BW,
              "Silver1.06" = 1.06 * s / n^(1/5),
              "Silver0.9" = 0.9 * s / n^(1/5),
              "Silver0.9IQR" = {
                iqr_val <- IQR(x)
                sigma <- min(s, iqr_val / 1.34)
                0.9 * sigma / n^(1/5)
              })
  bw_method <- BW
} else if (is.numeric(BW) && length(BW) == 1 && BW > 0) {
  h <- BW
  bw_method <- "User-defined"
} else {
  stop("BW must be numeric > 0 or one of: 'Silver1.06', 'Silver0.9', 'Silver0.9IQR'.")
}
kde <- density(x, bw = h, n = 2^20)
fx <- kde$y
x_vals <- kde$x
dx <- diff(x_vals)[1]
Fx <- cumsum(fx) * dx
Fx <- Fx / max(Fx)
inv_cdf <- approxfun(Fx, x_vals, rule = 2)
P0.5 <- inv_cdf(0.5)
P0.99865 <- inv_cdf(0.99865)

Ppu <- (usl_val - P0.5) / (P0.99865 - P0.5)
```

```
  df_result <- data.frame(
    Ppu = round(Ppu, 3),
    P0.5 = round(P0.5, 3),
    P0.99865 = round(P0.99865, 3),
    N = n,
    Sample_SD = round(s, 3),
    Bandwidth_Method = bw_method,
    Bandwidth_Value = round(h, 5)
  )

  return(df_result)
}
```

5.2 Tradition Ppu for each subgroup and the minmal as the CQA's Ppu

Ppu_SWWC_KDEDP function digests ONE data set such as DAR, CAR, or Mic (but not all of them). It automatically divides the data into subgroups by USL and then calculate each subgroups Ppu and take the minimal as this data set Ppu.

```
Ppu_SWWC_KDEDP <- function(data, Residue, USL, BW = "Silver1.06") {
  library(dplyr)
  library(rlang)

  residue_quo <- enquo(Residue)
  usl_quo <- enquo(USL)

# Filter rows with non-NA Residue and USL, and select those columns renamed to fixed names
data_clean <- data %>%
  filter(!is.na(!!residue_quo), !is.na(!!usl_quo)) %>%
  select(Residue = !!residue_quo, USL = !!usl_quo)

  if (nrow(data_clean) == 0) {
    stop("No valid rows after removing NA values in Residue or USL.")
  }

  subgroups <- split(data_clean, data_clean$USL)

  ppu_named_vector <- sapply(names(subgroups), function(usl_val) {
```

```
    sub_df <- subgroups[[usl_val]]
    res_values <- sub_df$Residue

    if (length(unique(res_values)) == 1) {
      return(100)
    } else {
  # For the following, we used USL-normalization method. In fact, it can be replaced by PP
  # The output is the same since Ppu is an invairant under USL-normalization
  result <- Ppu_KDEDPonUSLND(sub_df, Residue = Residue, USL = USL, BW = BW)
      return(result$Ppu)
    }
  })

  names(ppu_named_vector) <- paste0("Ppu_USL_", names(ppu_named_vector))

  overall_min <- min(ppu_named_vector, na.rm = TRUE)
  output <- as.data.frame(as.list(c(Ppu = overall_min, ppu_named_vector)), stringsAsFactor

  return(output)
}
```

5.3 Tradition Ppu for each subgroup and the minmal as the 1 to 3 CQAs' Ppu

Ppu_SWWC_KDEPDE_Overall digests one to three data sets such as DAR, CAR, or Mic.
If we input one dataset, it is the same as the output of Ppu_SWWC_KDEDP.

```
Ppu_SWWC_KDEDP_Overall <- function(
    data1, Residue1, USL1,
    data2 = NULL, Residue2 = NULL, USL2 = NULL,
    data3 = NULL, Residue3 = NULL, USL3 = NULL,
    BW = "Silver1.06"
) {
  library(rlang)
  Residue1_quo <- enquo(Residue1)
  USL1_quo <- enquo(USL1)

  Residue2_quo <- if (!missing(Residue2)) enquo(Residue2) else NULL
  USL2_quo <- if (!missing(USL2)) enquo(USL2) else NULL
```

```
  Residue3_quo <- if (!missing(Residue3)) enquo(Residue3) else NULL
  USL3_quo <- if (!missing(USL3)) enquo(USL3) else NULL

  valid_input <- function(data, residue_quo, usl_quo) {
    !is.null(data) && !quo_is_null(residue_quo) && !quo_is_null(usl_quo)
  }

  all_ppus <- c()

  if (valid_input(data1, Residue1_quo, USL1_quo)) {
    ppus1 <- Ppu_SWWC_KDEDP(data1, !!Residue1_quo, !!USL1_quo, BW = BW)
    names(ppus1) <- paste0("Data1_", names(ppus1))
    all_ppus <- c(all_ppus, as.list(ppus1))
  }

  if (valid_input(data2, Residue2_quo, USL2_quo)) {
    ppus2 <- Ppu_SWWC_KDEDP(data2, !!Residue2_quo, !!USL2_quo, BW = BW)
    names(ppus2) <- paste0("Data2_", names(ppus2))
    all_ppus <- c(all_ppus, as.list(ppus2))
  }

  if (valid_input(data3, Residue3_quo, USL3_quo)) {
    ppus3 <- Ppu_SWWC_KDEDP(data3, !!Residue3_quo, !!USL3_quo, BW = BW)
    names(ppus3) <- paste0("Data3_", names(ppus3))
    all_ppus <- c(all_ppus, as.list(ppus3))
  }

  if (length(all_ppus) == 0) {
    stop("No valid datasets or variables provided.")
  }

  subgroup_ppus <- all_ppus[!grepl("^Ppu$", names(all_ppus))]
  overall_min <- min(unlist(subgroup_ppus), na.rm = TRUE)

  output_vector <- c(Ppu_Overall = overall_min, subgroup_ppus)
  output_df <- as.data.frame(as.list(output_vector), stringsAsFactors = FALSE)

  return(output_df)
}
```

Table 1 is Table 2 of the Manuscript, and the same logic for other tables.

6.1 Ppu of the three subgroups of DAR by USL normalization

They are equal to each subgrop's Ppu in the following section.

```
DAR122_KDEDPonUSLND <- Ppu_KDEDPonUSLND(data=Eq_DAR_122, Residue=DAR, USL=USL)
colnames(DAR122_KDEDPonUSLND ) <- paste0("**", colnames(DAR122_KDEDPonUSLND ), "**")
kable(DAR122_KDEDPonUSLND )
```

Table 1: Ppu of Subgroup of DAR with USL=122.2 ug/swab Using USL-normalization and USL_Pct=100

Ppu	P0.5	P0.99865	N	Sample_SD	Bandwidth_Method	Bandwidth_Value
2.159	3.041	47.947	18	10.375	Silver1.06	6.16928

```
DAR122_KDEDP <- Ppu_KDEDP(data=Eq_DAR_122, Residue=DAR, USL=USL)
colnames(DAR122_KDEDP ) <- paste0("**", colnames(DAR122_KDEDP ), "**")
kable(DAR122_KDEDP )
```

Table 2: Ppu of Subgroup of DAR with USL=122.2 ug/swab by KDEDP Using USL=122.2

Ppu	P0.5	P0.99865	N	Sample_SD	Bandwidth_Method	Bandwidth_Value
2.159	3.716	58.591	18	12.678	Silver1.06	7.53886

```
DAR169_KDEDPonUSLND <- Ppu_KDEDPonUSLND(data=Eq_DAR_169, Residue=DAR, USL=USL)
colnames(DAR169_KDEDPonUSLND ) <- paste0("**", colnames(DAR169_KDEDPonUSLND ), "**")
kable(DAR169_KDEDPonUSLND )
```

Table 3: Ppu of Subgroup of DAR with USL=169.7 Using USL-normalization and USL_Pct=100

Ppu	P0.5	P0.99865	N	Sample_SD	Bandwidth_Method	Bandwidth_Value
65.515	0.318	1.839	24	0.391	Silver1.06	0.21956

```
DAR169_KDEDP <- Ppu_KDEDP(data=Eq_DAR_169, Residue=DAR, USL=USL)
colnames(DAR169_KDEDP ) <- paste0("**", colnames(DAR169_KDEDP ), "**")
kable(DAR169_KDEDP )
```

Table 4: Ppu of Subgroup of DAR with USL=169.7 ug/swab by KDEDP Using USL=169.7

Ppu	P0.5	P0.99865	N	Sample_SD	Bandwidth_Method	Bandwidth_Value
65.515	0.539	3.121	24	0.664	Silver1.06	0.37259

```
DAR20_KDEDPonUSLND <- Ppu_KDEDPonUSLND(data=Eq_DAR_20, Residue=DAR, USL=USL)
colnames(DAR20_KDEDPonUSLND ) <- paste0("**", colnames(DAR20_KDEDPonUSLND ), "**")
kable(DAR20_KDEDPonUSLND )
```

Table 5: Ppu of Subgroup of DAR with USL=20 ug/swab Using USL-normalization and USL_Pct=100

Ppu	P0.5	P0.99865	N	Sample_SD	Bandwidth_Method	Bandwidth_Value
23.414	1.854	6.046	18	1.473	Silver1.06	0.87567

```
DAR20_KDEDP <- Ppu_KDEDP(data=Eq_DAR_20, Residue=DAR, USL=USL)
colnames(DAR20_KDEDP ) <- paste0("**", colnames(DAR20_KDEDP ), "**")
kable(DAR20_KDEDP )
```

Table 6: Ppu of Subgroup of DAR with USL=20 ug/swab by KDEDP Using USL=20

Ppu	P0.5	P0.99865	N	Sample_SD	Bandwidth_Method	Bandwidth_Value
23.414	0.371	1.209	18	0.295	Silver1.06	0.17513

6.2 Ppu of DAR by the minimal of each subgroup (subgroup wise worst case)

```
SWWCDAR <- Ppu_SWWC_KDEDP(data=Eq_DAR, Residue=DAR, USL=USL)
colnames(SWWCDAR) <- paste0("**", colnames(SWWCDAR), "**")
kable(SWWCDAR)
```

Table 7: The SWWC Ppu of DAR and Ppu of Each Subgroup of DAR

Ppu	Ppu_USL_20	Ppu_USL_122.2	Ppu_USL_169.7
2.159	23.414	2.159	65.515

#6.3 Ppu of DAR by pooling USL normalized data

```
SWWC_KDEDPonUSLND_DAR <- Ppu_KDEDPonUSLND(data=Eq_DAR, Residue=DAR, USL=USL)
colnames(SWWC_KDEDPonUSLND_DAR ) <- paste0("**", colnames(SWWC_KDEDPonUSLND_DAR ), "**")
kable(SWWC_KDEDPonUSLND_DAR )
```

Table 8: Ppu of DAR by Pooling the Three Subgroups with USL-Normalization

Ppu	P0.5	P0.99865	N	Sample_SD	Bandwidth_Method	Bandwidth_Value
2.568	1.339	39.753	60	6.044	Silver1.06	2.82492

#6.3 Ppu of CAR by USL-normalization

```
SWWC_KDEDPonUSLND_CAR <- Ppu_KDEDPonUSLND(data=Eq_CAR, Residue=CAR, USL=USL)
colnames(SWWC_KDEDPonUSLND_CAR ) <- paste0("**", colnames(SWWC_KDEDPonUSLND_CAR ), "**")
kable(SWWC_KDEDPonUSLND_CAR )
```

Table 9: Ppu of CAR

Ppu	P0.5	P0.99865	N	Sample_SD	Bandwidth_Method	Bandwidth_Value
26.348	1.278	5.025	33	0.835	Silver1.06	0.44

7 Ppu of Mic by USL-normalization

```
SWWC_KDEDPonUSLND_Mic <- Ppu_KDEDPonUSLND(data=Eq_Mic, Residue=Mic, USL=USL)
colnames(SWWC_KDEDPonUSLND_Mic ) <- paste0("**", colnames(SWWC_KDEDPonUSLND_Mic ), "**")
kable(SWWC_KDEDPonUSLND_Mic )
```

Table 10: Ppu of Mic

Ppu	P0.5	P0.99865	N	Sample_SD	Bandwidth_Method	Bandwidth_Value
20.164	0.035	4.993	20	0.894	Silver1.06	0.52077

8.1 Overall Ppu by SWWC (Subgroup wise worst case)

```
SWWC_KDEDP_Overall <- Ppu_SWWC_KDEDP_Overall(
    data3=Eq_Mic, Residue3=Mic, USL3=USL,
    data2 = Eq_CAR, Residue2 = CAR, USL2 = USL,
    data1 = Eq_DAR, Residue1 = DAR, USL1 = USL,
    BW = "Silver1.06")
colnames(SWWC_KDEDP_Overall ) <- paste0("**", colnames(SWWC_KDEDP_Overall ), "**")
kable(
  SWWC_KDEDP_Overall)
```

Table 11: Process Overall Ppu by SWWC_KDEDP

Ppu_Overall	Data1_Ppu	Data1_Ppu_USL_20	Data1_Ppu_USL_25	Data2_Ppu	Data2_Ppu_USL_20	Data2_Ppu_USL_25	Data3_Ppu	Data3_Ppu_USL_25
2.159	2.159	23.414	2.159	65.515	26.348	26.348	20.164	20.164

8.2 Overall Ppu by CQAWWC (CQA wise worst case)

```
CQAWWC_KDEDPonUSLND_Overall <- Ppu_CQAWWC_KDEDPonUSLND(
    data3=Eq_Mic, Residue3=Mic, USL3=USL,
    data2 = Eq_CAR, Residue2 = CAR, USL2 = USL,
    data1 = Eq_DAR, Residue1 = DAR, USL1 = USL,
    BW = "Silver1.06")
colnames(CQAWWC_KDEDPonUSLND_Overall ) <- paste0("**", colnames(CQAWWC_KDEDPonUSLND_Overal
kable(CQAWWC_KDEDPonUSLND_Overall)
```

Table 12: Process Overall Ppu by CQAWWC_KDEDPonUSLND

Ppu	Ppu_Data1	Ppu_Data2	Ppu_Data3
2.568	2.568	26.348	20.164

8.3 Overall Ppu by CQAWP (CQA wise pooling)

```
CQAWP_KDEDPonUSLND_Overall <- Ppu_CQAWP_KDEDPonUSLND(
    data3=Eq_Mic, Residue3=Mic, USL3=USL,
    data2 = Eq_CAR, Residue2 = CAR, USL2 = USL,
    data1 = Eq_DAR, Residue1 = DAR, USL1 = USL,
    BW = "Silver1.06")
colnames(CQAWP_KDEDPonUSLND_Overall ) <- paste0("**", colnames(CQAWP_KDEDPonUSLND_Overall
kable(CQAWP_KDEDPonUSLND_Overall)
```

Table 13: Process Overall Ppu by CQAWP_KDEDPonUSLND

Ppu	P0.5	P0.99865	N	Sample_SD	Bandwidth_Meth	Bandwidth_Value
2.699	1.067	37.719	113	4.506	Silver1.06	1.85541

9 CQAWWC Model Monitoring Cleaning Process Performance

```
CQAWWC_Model <- CQAWWC_BAKDEDPonUSLND_CVStage3Monitoring(data1=Eq_DAR, Residue1=DAR, USL1=
data3=Eq_Mic, Residue3=Mic, USL3=USL,Fiscal_Year3=Fiscal_Year,Time_cut=2025, initial_boot
colnames(CQAWWC_Model) <- paste0("**", colnames(CQAWWC_Model), "**")
kable(CQAWWC_Model)
```

Table 14: Monitoring Results of the Cleaning Process for Equipment A by CQAWP-BAKDEDPonUSLND Model

Ppu_training	Ppu_threshold	Ppu_monitoring	decision
2.528	2.304	2.568	Cleaning process is capable.

10 CQAWP Model Monitoring Cleaning Process Performance

```
CQAWP_Model <- CQAWP_BAKDEDPonUSLND_CVStage3Monitoring(data1=Eq_DAR, Residue1=DAR, USL1=US
data3=Eq_Mic, Residue3=Mic, USL3=USL,Fiscal_Year3=Fiscal_Year,Time_cut=2025, initial_boot
colnames(CQAWP_Model) <- paste0("**", colnames(CQAWP_Model), "**")

kable(CQAWP_Model )
```

Table 15: Monitoring Results of the Cleaning Process for Equipment A by CQAWP-BAKDEDPonUSLND Model

Ppu_training	**Ppu_threshold**	**Ppu_monitoring**	**Performance_conclusion**
2.676	2.497	2.699	Cleaning process is capable.