Modified ALCC

Adrian A. Correndo

03/22/2020

This code was prepared as a tutorial for potential users of the Modified Arcsine-log Calibration Curve (ALCC) detailed in Correndo et al. (2017).

Instructions for users

1. Load your soil test value (STV) and relative yield (RY) data as vectors of a data.frame.
2. Specify into the function -ALCC()- which vectors correspond to RY and STV.

Note: They new update allows to specify “data” argument as optional.

3. Specify your relative yield target (e.g. 90%) and desired confidence level (e.g. 0.95 for 1 - alpha(0.05)). Used for the estimation of critical soil test value (CSTV) and lower and upper confidence limits.
4. Run.
5. Check results and adjust plot as desired.
6. Please, refer any question to Adrian Correndo, correndo@agro.uba.ar - correndo@ksu.edu.

Note: RY should be expressed relative to a maximum in order to obtain values bounded at %100. Otherwise, arcsine transformation doesn’t work. If RY values > 100% are found, the fucntion will cap them up to 100% and will display a warning about this.

References

Correndo, A.A., F. Salvagiotti, F.O. García, y F.H. Gutiérrez-Boem. 2017. A modification of the arcsine-log calibration curve for analysing soil test value - relative yield relationships. Crop & Pasture Science 68 (3): 297-304, doi: 10.10171/CP16444

Last update: 03-22-2022

1. Libraries

```r
# Install if needed
# install.packages("easypackages")
# install.packages("devtools")
library(easypackages) # Helps to load packages and install & load them if they are not installed yet.
library(devtools)
packages("readxl") # Open xlsx files
packages("tidyverse", "ggpmisc") # Data wrangling and plots
packages("smatr") # SMA regression analysis for reference
packages("agridat") # SMA regression analysis for reference
```

2. Data

```r
# Example 1 dataset
data_1 = data.frame("RY" = c(65,80,85,88,90,94,93,96,97,95,98,100,99,99,100),
                   "STV" = c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15))


# Example 2 dataframe. Imported from csv file
# It can be easily replaced with your own csv file
data_2 = read.csv(file = "data_test.csv")

# Example 3 dataframe. Imported from xlsx file
data_3 = readxl::read_xlsx(path ="data_3.xlsx", sheet = 1)

# Create nested structure as example of multiple datasets
data.all = bind_rows(data_1, data_2, data_3, .id = "id") %>%
  tidyr::nest(data = c("STV", "RY"))
```

3. ALCC function, updated

```r
ALCC <- function(data=NULL, RY, STV, target, confidence){

  rlang::eval_tidy(data = data,
                   rlang::quo(
  if (max({{RY}}) > 100) {
        warning("One or more original RY values exceeded 100%.\nAll RY values greater than 100% have bee
    } ) )

  # Add a function to cap if there are RY values > 100
  ry <- rlang::eval_tidy(data = data, rlang::quo(ifelse({{RY}} > 100, 100, as.double({{RY}})) ))
  n <- length(ry) # Sample size
  df <- n - 2 # Degrees of freedom
  prob <- 1-((1-confidence)/2) # Probability for t-dist
  tvalue <- qt(p=prob, df = df) # Student-t value
  arc_RY <- asin(sqrt(ry/100)) - asin(sqrt(target/100)) # RY transformation (centered to target)
  ln_STV <- rlang::eval_tidy(data = data, rlang::quo(log({{STV}}) ) ) # STV natural log transformation
  r <- cor(ln_STV, arc_RY, method = "pearson") # Pearson correlation (r)
  p_value <- cor.test(ln_STV,arc_RY, method = "pearson")$p.value # p-value of r
  slope <- sd(ln_STV)/sd(arc_RY) # SMA slope for ln_STV ~ arc_RY
  intercept <- mean(ln_STV) - (mean(arc_RY)*slope) # Intercept
  Line <- intercept + slope * arc_RY # Fitted ln_STV for observed RY
  CSTV <- exp(intercept) # Critical STV for specified RY-target and confidence (1-alpha)
  MSE <- sum((Line-ln_STV)^2)/df # Mean Square Error of ln_STV
  SSx <- sum((mean(arc_RY)-arc_RY)^2)  # Sum of Squares of arc_RY
  SE_int <- sqrt(MSE*((1/n)+ ((mean(arc_RY)^2)/SSx)))  # Standard Error intercept
  CSTV_lower <- exp(intercept - (tvalue * SE_int))  # Lower limit of CSTV
  CSTV_upper <- exp(intercept + (tvalue * SE_int)) # Upper limit of CSTV
  new_RY <- seq(min(ry),100, by=0.2) # New RY vector up to %100 to fit curve
  new_arc_RY <- asin(sqrt(new_RY/100)) - asin(sqrt(target/100)) # Transforming new_RY vector
  fitted_Line <- intercept + slope * new_arc_RY # Fitted ln_STV for new_RY
  fitted_STV <- exp(fitted_Line) # Fitted ln_STV for new_RY
  target <- target
  confidence <- confidence

  # Outcome
  results <- as.data.frame(list("n" = n,
                                "r" = r,
                                "p_value" = p_value,
                                "target" = target,
                                "confidence" = confidence,
                                "CSTV" = CSTV,
                                "LL" = CSTV_lower,
                                "UL" = CSTV_upper,
                                "RY.fitted" = new_RY,
                                "STV.fitted" = fitted_STV))   %>%
    tidyr::nest(Curve = c("RY.fitted", "STV.fitted"))


  return(results)
}
```

4. Fit examples

4.1. Fit ALCC models individually

```
# RY target = 90%, confidence level = 0.95, replace with your desired values

# Data 1
# Using dataframe
fit_example_1 = ALCC(data = data_1, RY = RY, STV = STV, target=90, confidence = 0.95)
# Alternative using the vectors
# fit_example_1 = ALCC(RY = data_1$RY,STV = data_1$STV, target=90,confidence = 0.95)

fit_example_1
```

```
## # A tibble: 1 x 9
##       n     r       p_value target confidence  CSTV    LL    UL Curve
##   <int> <dbl>         <dbl>  <dbl>      <dbl> <dbl> <dbl> <dbl> <list>
## 1    15 0.968 0.00000000330     90       0.95  4.48  3.95  5.08 <tibble>
```

```
# Data 2
fit_example_2 = ALCC(data = data_2, RY = RY, STV = STV, target=90, confidence = 0.95)

fit_example_2
```

```
## # A tibble: 1 x 9
##       n     r  p_value target confidence  CSTV    LL    UL Curve
##   <int> <dbl>    <dbl>  <dbl>      <dbl> <dbl> <dbl> <dbl> <list>
## 1   137 0.716 7.31e-23     90       0.95  23.3  21.6  25.1 <tibble [441 x 2]>
```

```
# Data 3
fit_example_3 = ALCC(data = data_3, RY = RY, STV = STV, target=90, confidence = 0.95)

fit_example_3
```

```
## # A tibble: 1 x 9
##       n     r   p_value target confidence  CSTV    LL    UL Curve
##   <int> <dbl>     <dbl>  <dbl>      <dbl> <dbl> <dbl> <dbl> <list>
## 1   107 0.374 0.0000741     90       0.95  21.8  19.4  24.5 <tibble [376 x 2]>
```

4.2. Fit multiple ALCC models with mapping

```
# Run multiple examples at once with map()
fit_examples = data.all %>%
  mutate(modALCC = map(data, ~ALCC(RY = .$RY, STV = .$STV, target=90, confidence = 0.95))) %>%
  unnest(., cols = c("modALCC"))

head(fit_examples)
```

```
## # A tibble: 3 x 11
##   id    data          n     r  p_value target confidence  CSTV    LL    UL
##   <chr> <list>    <int> <dbl>    <dbl>  <dbl>      <dbl> <dbl> <dbl> <dbl>
## 1 1     <tibble>     15 0.968 3.30e- 9     90       0.95  4.48  3.95  5.08
## 2 2     <tibble>    137 0.716 7.31e-23     90       0.95 23.3   21.6  25.1 
## 3 3     <tibble>    107 0.374 7.41e- 5     90       0.95 21.8   19.4  24.5 
## # ... with 1 more variable: Curve <list>
```

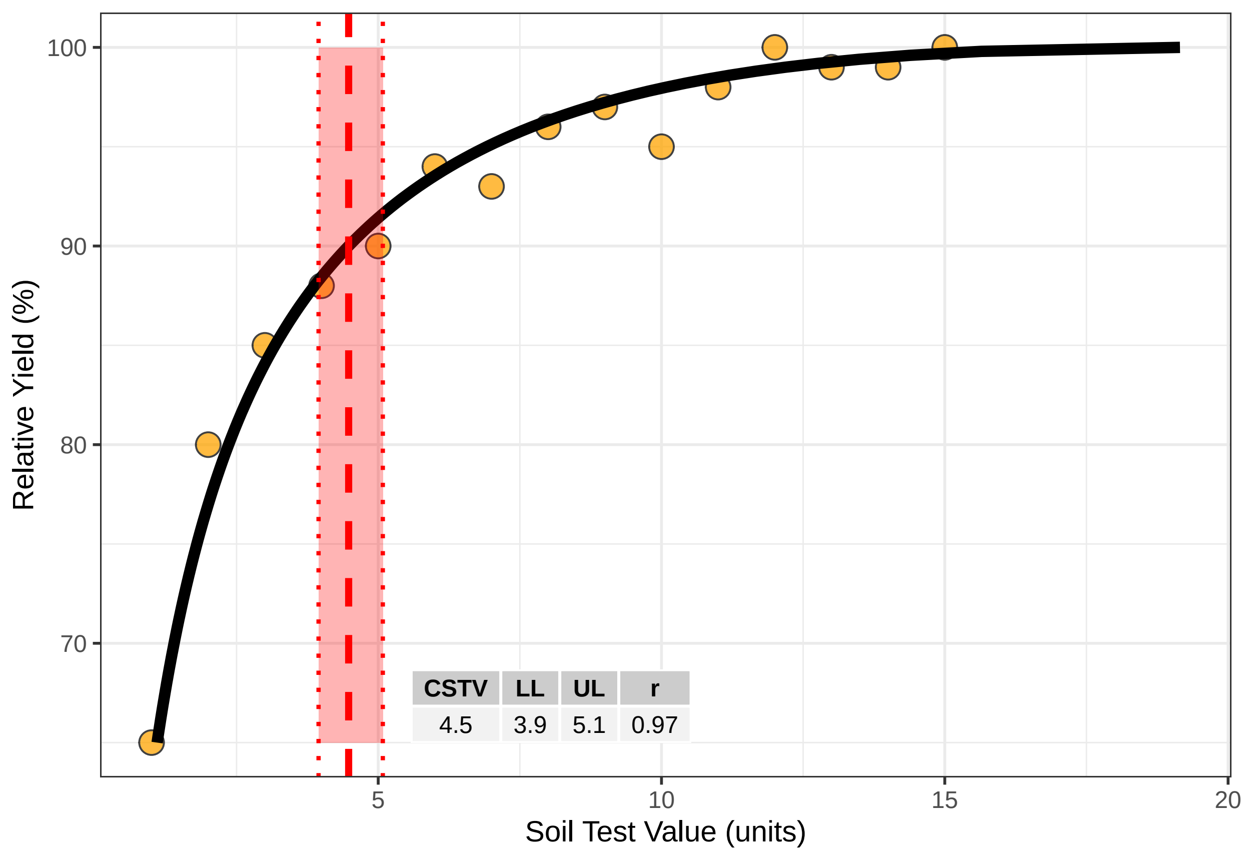
```
# Alternative with group_map, this does not required nested data.
fit_all = bind_rows(data_1, data_2, data_3, .id = "id") %>%
  group_by(id) %>%
  group_map(~ALCC(data = ., RY = RY, STV = STV, target = 90, confidence = 0.95))

head(fit_all)
```

```
## [[1]]
## # A tibble: 1 x 9
##       n     r       p_value target confidence  CSTV    LL    UL Curve   
##   <int> <dbl>         <dbl>  <dbl>      <dbl> <dbl> <dbl> <dbl> <list>  
## 1    15 0.968 0.00000000330     90       0.95  4.48  3.95  5.08 <tibble>
## 
## [[2]]
## # A tibble: 1 x 9
##       n     r  p_value target confidence  CSTV    LL    UL Curve             
##   <int> <dbl>    <dbl>  <dbl>      <dbl> <dbl> <dbl> <dbl> <list>            
## 1   137 0.716 7.31e-23     90       0.95  23.3  21.6  25.1 <tibble [441 x 2]>
## 
## [[3]]
## # A tibble: 1 x 9
##       n     r   p_value target confidence  CSTV    LL    UL Curve             
##   <int> <dbl>     <dbl>  <dbl>      <dbl> <dbl> <dbl> <dbl> <list>            
## 1   107 0.374 0.0000741     90       0.95  21.8  19.4  24.5 <tibble [376 x 2]>
```

5. Plots

5.1. Example 1

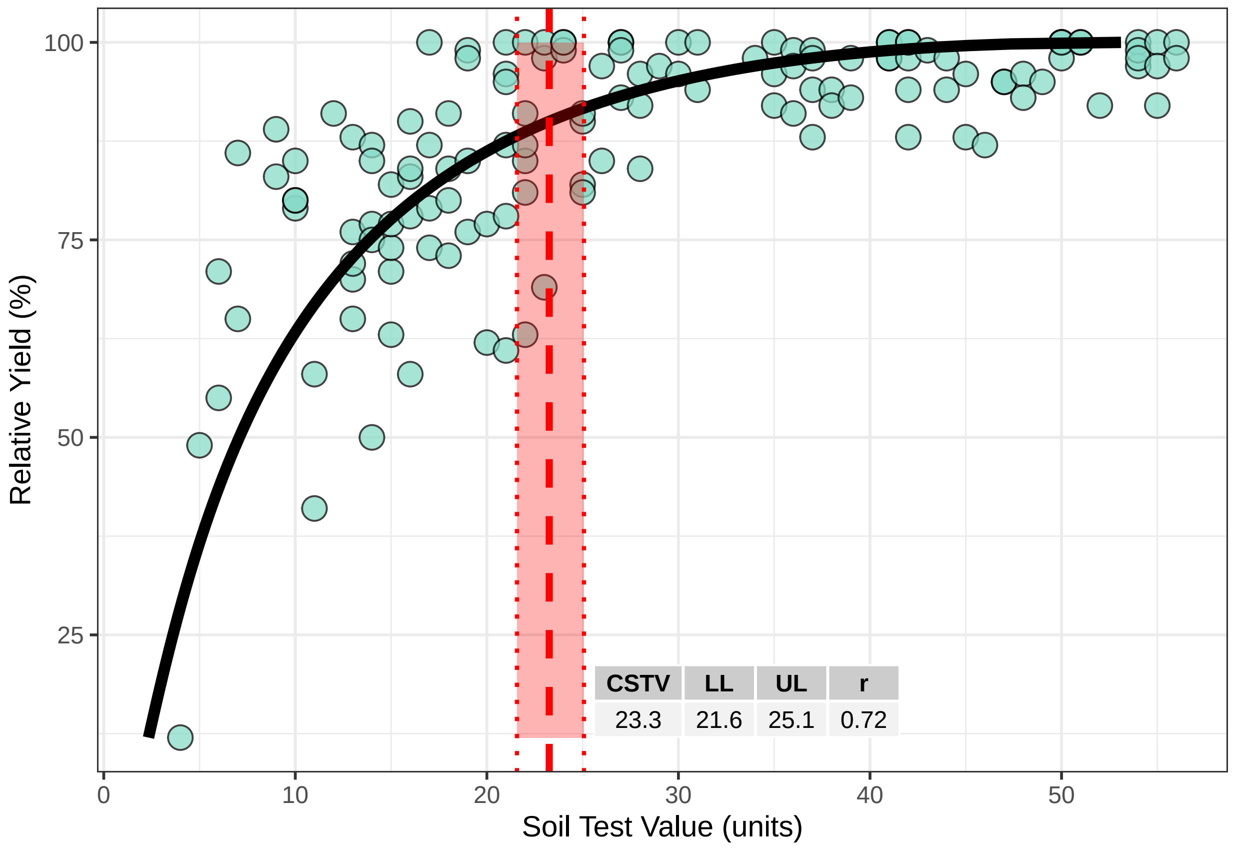
```r
# Extracting curve data as a data.frame to plot
curve_example1 = fit_example_1 %>% unnest(., cols = Curve)

# Plot
data_1 %>% ggplot()+
  # Points
  geom_point(aes(x = STV, y = RY), fill = "orange", shape = 21, size = 4, alpha = 0.75)+
  # Fitted ALCC
  geom_line(data = curve_example1, aes(x= STV.fitted, y = RY.fitted), size = 2)+
  # Critical value
  geom_vline(xintercept = fit_example_1$CSTV, col = "red", size = 1.25, linetype = "dashed")+
  # Confidence limits
  # Lines
  geom_vline(xintercept = fit_example_1$LL, col = "red", size = 0.75, linetype = "dotted")+
  geom_vline(xintercept = fit_example_1$UL, col = "red", size = 0.75, linetype = "dotted")+
  # Shade
  ggpp::annotate(geom = "rect", xmin = fit_example_1$LL, xmax = fit_example_1$UL,
                 ymin = min(data_1$RY), ymax = 100, alpha = .3, fill = "red")+
  # Axis titles
  labs(x = "Soil Test Value (units)", y = "Relative Yield (%)")+
  theme_bw()+
  theme()+
  # Annotate critical values data
  ggpp::annotate(geom = "table", y = min(data_1$RY), x = fit_example_1$UL + 0.5, hjust= 0, vjust = 0,
                 label = fit_example_1 %>% dplyr::select(CSTV, LL, UL, r) %>%
                   mutate_at(.vars = c("r"), ~round(.,2)) %>%
                   mutate_at(.vars = c("CSTV","LL","UL"), ~round(.,1))
                   )
```



CSTV	LL	UL	r
4.5	3.9	5.1	0.97

5.2. Example 2

```
# Extracting curve data as a data.frame to plot
curve_example2 = fit_example_2 %>% unnest(., cols = Curve)

# Plot
data_2 %>% ggplot()+
  # Points
  geom_point(aes(x = STV, y = RY), fill = "#88dbc8", shape = 21, size = 4, alpha = 0.75)+
  # Fitted ALCC
  geom_line(data = curve_example2, aes(x= STV.fitted, y = RY.fitted), size = 2)+
  # Critical value
  geom_vline(xintercept = fit_example_2$CSTV, col = "red", size = 1.25, linetype = "dashed")+
  # Confidence limits
  geom_vline(xintercept = fit_example_2$LL, col = "red", size = 0.75, linetype = "dotted")+
  geom_vline(xintercept = fit_example_2$UL, col = "red", size = 0.75, linetype = "dotted")+
  ggpp::annotate(geom = "rect", xmin = fit_example_2$LL, xmax = fit_example_2$UL,
                 ymin = min(data_2$RY), ymax = 100, alpha = .3, fill = "red")+
  # Axis titles
  labs(x = "Soil Test Value (units)", y = "Relative Yield (%)")+
  theme_bw()+
  theme()+
  # Annotate critical values data
  ggpp::annotate(geom = "table", y = min(data_2$RY), x = fit_example_2$UL + 0.5, hjust= 0, vjust = 0,
                 label = fit_example_2 %>% dplyr::select(CSTV, LL, UL, r) %>%
                   mutate_at(.vars = c("r"), ~round(.,2)) %>%
                   mutate_at(.vars = c("CSTV","LL","UL"), ~round(.,1))
                   )
```



CSTV	LL	UL	r
23.3	21.6	25.1	0.72

6. Revised SoilTestCocaCola Code

Great initiative from Dr. Austin Pearce working for automatize critical nutrient values estimations for the FRST (Fertilizer Recommendation System Tool) Project.

Original source: https://github.com/austinwpearce/SoilTestCocaCola.git

Copyright (c) 2022 Austin Pearce

Adapted by Adrian Correndo on 03/22/2022

List of modifications:

1. Modified variables (e.g. x, xt_centered, y, yt) and arguments names (inside functions; e.g. sufficiency for target) for more familiar terminology I personally use.

i. For example, replaced "lower_cl" with "lower_cstv" to maintain consistency.

2. Removed sma argument from formulae. Not needed as the modified ALCC is entirely based on SMA. OLS steps described by Correndo et al. (2017) are just some tricks to obtain parameters using a spreadsheet where we cannot run a sma() model (Warton et al., 2006).
3. Modified slope formula for more explicit one that removes the need of estimating an OLS slope and then rotate. New formula is simply the ratio between standard deviations of variables, as follows:

i. slope = sd(ln_STV) / sd(arc_ry)

4. Shortened some steps regarding centering variables.
5. Fixed model pvalue (replaced student-t pvalue for Pearson-r pvalue)
6. Modified df as n-2 for the t-test. (see Table 4, Warton et al., 2006)
7. Modified test_stat formula, obtained directly from qt() instead of pt()
8. Added probability formula (prob), which includes confidence to later estimate the t_stat.
9. Shortened se formula, obtained directly from qt() instead of pt()
10. Modified ssx formula for more explicit one
11. Fixed model pvalue (replaced student-t pvalue for Pearson-r pvalue)
12. Fixed issue with dplyr::if_else() when capping RY to 100%
13. Modified data points removal criteria (remove2x replaced by remove.leverage, which gives users more autonomy)

i. Users can choose now between options remove.leverage = c("none", "cstv90_2x", "cstv100")
ii. Added sample size (n) used for CSTV calculations as outcome to show when removing points.
iii. Modified printed text in the alcc_plot() in accordance with the new remove.leverage criteria.

14. Added some extra warning regarding the specification of relative yield "target" and "confidence". Although defaults are provided, I consider important to create awareness on users about this, which are key features the ALCC model brings.
15. By the end of stage_3 of alcc(), I replaced "mutate(remove2x =”TRUE")" with "mutate(remove.leverage = remove.leverage)", otherwise the summary table always printed "TRUE" even though STV > cstv90_2x were NOT removed (remove2x = FALSE). It currently prints the user specification ("none", "cstv90_2x", "cstv100").

6.1. Revised alcc() functions

```r
# ALCC
# =============================================================================
alcc_core.rev <- function(data,
                  # # AC: I would rather use the acronym of soil test value (stv),
                    # as the same is done for "ry".
                    #stv,
                    stv,
                    ry,
    # AC: WARNING: There is no really an option for "SMA" or not, it is ALWAYS SMA.
                    # sma,
    # AC: I think the word sufficiency is not appropriate for the case...
      # most likely this is just a "target" RY value
      # sufficiency sounds like 100%, a target would be just an "acceptable" RY
                    target,
    # AC: I added confidence as a needed arg. to define
                    confidence) {

    if (nrow(data) < 8) {
        stop("Too few data points. Try at least 8.")
    }

    if (missing(stv)) {
        stop("Specify the column containing soil test values (e.g., stv = STK)")
    }

    if (missing(ry)) {
        stop("Specify the column containing relative yield (%) values (e.g., ry = RY)")
    }
# AC: # Add statement for missing target, optional since a DEFAULT is provided (90)
    if (missing(target)) {
        stop("Specify the target value of relative yield (%) (e.g., target = 90)")
    }

 # AC: # Add statement for missing confidence, optional since a DEFAULT is provided (95)
    if (missing(confidence)) {
        stop("Specify the desired confidence level (%) for cstv estimation (e.g., confidence = 95)")
    }


    # enquo let's the user enter their own column names for x and y variables
# AC: I think this is particularly confusing because in the SMA regression
# Soil test is not in the x-axis but on the y-axis,
# I would rather use "STV" instead of "x", and then you flip them for transformed (xt, yt)
    #x <- enquo(soil_test)
    STV <- enquo(stv)
 # AC: for SMA regression RY is not in the y-axis but on the x-axis,
  # I would rather use "STV" instead of "x"
    RY <- enquo(ry)
    #y <- enquo(ry)

    steps_1_4 <- data %>%
        mutate(
            stv = !!STV,
            # RY values greater than 100 are capped at 100
            ry_cap = if_else(!!RY > 100, 100, as.double(!!RY)),
            # Tranform back to numeric, just in case
            ry_cap = as.numeric(ry_cap),
      # AC: Model is not necessary, it is always SMA
            #model = case_when(sma == TRUE ~ "ALCC-SMA",
            #                  sma == FALSE ~ "ALCC"),
            # get sample size
            n = nrow(data),
      # AC: added degrees of freedom of the model with two pars
      # (See Warton et al. 2006, Table 4)
            # n-1 was used
            df = n-2,
            # Step 1 Transform (t for "transformed" added to x and y)
      # AC: ### THIS IS NOT ADDED TO X AND Y, note you have flipped them...
      # Thus, I would avoid using x & y
            # for ALCC, soil test goes to Y-axis and RY goes to X-axis
            # the names of variables are difficult to follow, at least for me,
      # AC: I would suggest use ln_STV and arc_RY
            ln_stv = log(stv),
            # Step 2 Center
      # AC: I would center right here as well
            arc_ry = asin(sqrt(ry_cap / 100)) - asin(sqrt(target/100)),
            target = target, # I think its not necessary
      # AC: UNNECESSARY
            # adjust_by = asin(sqrt(sufficiency / 100)),
      ### NOT NEEDED, adjust_by = asin(sqrt(target / 100)),
            ### xt_centered = xt - adjust_by,
            # Step 3 Correlation (Pearson)
            r = cor(ln_stv, arc_ry, method = "pearson"),
      # AC: The student-t formula here DID NOT include the confidence level so it
            # was always estimated for 95% confidence level
              #t_stat  = (r * sqrt(df)) / sqrt(1 - r ^ 2),
      # AC: Warning: the df here are n-2, also this is informed later as the p-value
      #of the model
      # but this is not the pvalue of the model, it's just the pvalue of the t-test
      # (if B0 =/ from zero)
      # the pvalue of the model is the pvalue of the Pearson correlation test
            #pvalue  = pt(t_stat, df = n - 1, lower.tail = FALSE),
      # AC: I would replace with the following
            prob = 1-((1-confidence/100)/2), # Probability for t-dist
      # AC: The Student-t value from quantile fn (qt) saves a step for the LL
      #and UL estimations
            t_stat = qt(p=prob, df = df),
      # AC: This is the pvalue of the model
      # p-value of Pearson correlation (r)
            pvalue = stats::cor.test(ln_stv, arc_ry, method = "pearson")$p.value,
            # Step 4 Means
      # AC: We DON'T ACTUALLY NEED THE MEANS HERE, can be included into the intercept formula
                #mean_xt  = mean(xt_centered),
                #mean_yt  = mean(yt),
```

```r
        )


    # Step 5 Fit linear model to transformed, centered data
    # AC: # THIS STEP IS UNNECESSARY AS THE SMA-slope = sd(ln_stv)/ sd(arc_ry)
    # ols_center <- lm(yt ~ xt_centered, data = steps_1_4)

    steps_5_9 <- steps_1_4 %>%
        mutate(
            #intercept = coef(ols_center)[[1]],
            #slope = coef(ols_center)[[2]],
            # Step 6 Rotate the regression (SMA)
            # slope must come first
            # AC: I replaced the slope for the direct formula for SMA regression
            slope = sd(ln_stv) / sd (arc_ry),
            #slope = case_when(sma == TRUE ~ slope / pearson,
            #                  sma == FALSE ~ slope),
            # AC: I replaced the slope for the direct formula for SMA regression
            intercept = mean(ln_stv) - (mean(arc_ry)*slope),
            #intercept = case_when(sma == TRUE ~ mean_yt - slope * mean_xt,
            #                      sma == FALSE ~ intercept),
            # Step 7 Estimate Critical Soil Test Concentration
            cstv = exp(intercept),
            # Step 8 Estimate the confidence interval
            pred_ln_stv  = intercept + slope * arc_ry, # Fitted ln_STV for observed RY
            mse = sum((pred_ln_stv - ln_stv) ^ 2) / df,
            # AC: I would use a more explicit formula here and not this indirect way
            #ssx      = var(xt_centered) * (n - 1),
            ssx = sum((mean(arc_ry)-arc_ry)^2),
            confidence = confidence,
            se = sqrt(mse * ((1/n) + ((mean(arc_ry) ^ 2) / ssx))),
            #lower_cstv = exp(intercept - se * qt(1 - (1 - confidence / 100) / 2,
            #                                    df = n - 2)),
            #upper_cstv = exp(intercept + se * qt(1 - (1 - confidence / 100) / 2,
            #                                    df = n - 2)),
            # AC: I replaced boundaries formula for simpler ones
            lower_cstv = exp(intercept - (t_stat * se)), # Lower limit of CSTV
            upper_cstv = exp(intercept + (t_stat * se)), # Upper limit of CSTV
            # Step 9 Back-transform
            fitted_stv = exp(pred_ln_stv),
            fitted_ry = 100 * (sin(
                asin(sqrt(target/100)) + ((pred_ln_stv - intercept) / slope))) ^ 2
        ) %>%
        # 'dataset' might be problematic, not defined in scope
        select(target, cstv,
               lower_cstv, upper_cstv, confidence,
               fitted_stv, fitted_ry, pvalue, r, everything())
}

alcc.rev <- function(data,
                 stv,
                 ry,
                 #sma = TRUE,
                target = 90,
                confidence = 95,
                remove.leverage = "none",
                #remove2x = FALSE,
                summary = FALSE){

    if (missing(stv)) {
        stop("Specify the soil test variable (e.g., stv = STK)")
    }

    if (missing(ry)) {
        stop("Enter name of relative yield (%) variable (e.g., ry = RY)")
    }

    # AC: # Add statement for missing target
    if (missing(target)) {
        stop("Specify the target value of relative yield (%) (e.g., target = 90)")
    }

    # AC: # Add statement for missing confidence
    if (missing(confidence)) {
        stop("Specify the desired confidence level (%) for cstv estimation (e.g., confidence = 95)")
    }

    # enquo let's the user enter their own column names for x and y variables
    # AC: # I think this is particularly confusing because in the SMA regression
    # Soil test is not in the x-axis but on the y-axis, I will rather use "STV"
    # instead of "x"
    #x <- enquo(stv)
    STV <- enquo(stv)
    # AC: for SMA regression RY is not in the y-axis but on the x-axis, I will
    #rather use "STV" instead of "x"
    RY <- enquo(ry)
    #y <- enquo(ry)

    test <- data %>%
        select(!!STV, !!RY)

    if (max(test[,2]) > 100) {
        warning("One or more original RY values exceeded 100%.\nAll RY values greater than 100% have bee
    }

    if (min(test[,1]) < 0) {
        stop("One or more soil test values are less than 0. Remove all data with STV values less than 0
    }

    # stage 1, 2, and 3 described by Dyson and Conyers 2013
    stage_1 <- data %>%
        alcc_core.rev(
            stv = !!STV,
            ry = !!RY,
            # AC: sma arg. not needed.
            #sma = sma,
            target = 100,
            confidence = confidence
        )

    cstv_100 <<- unique(stage_1$cstv)

    # AC: Added a small note
    # identify CSTV for 90%
    stage_2 <- data %>%
    # AC: Warning! This filter line ALWAYS filter out stv > cstv100.
    # The user should be the one deciding. I don't agree with Dyson here,
    # as many others may share. The cstv90 estimate should be obtained
    # without removing anything.
    # All removes must be OPTIONAL and decision exclusive to researchers.
    # So the filter step should be at stage 3 only
        #filter(!!STV <= cstv_100) %>%
      alcc_core.rev(
            stv = !!STV,
            ry = !!RY,
      # AC: sma arg. not needed.
            #sma = sma,
            target = 90,
            confidence = confidence
        )

    # Extract cstv_90
    cstv90_2x <<- unique(stage_2$cstv) * 2

# AC: What if I don't want to remove soil test values > CSTV_100 ????

    stage_3 <- data %>%
# AC: I would suggest use 3 options as a string:
      #(i) "cstv90_2x", (ii) "cstv100", (iii) "none"
        #filter(case_when(remove2x == TRUE ~ !!STV <= cstv90_2x,
         #                 remove2x == FALSE ~ !!STV <= cstv_100)) %>%
      filter(case_when(remove.leverage == "none" ~ !!STV != 0,
                       remove.leverage == "cstv90_2x" ~ !!STV <= cstv90_2x,
                       remove.leverage == "cstv100" ~ !!STV <= cstv_100 ) ) %>%
        alcc_core.rev(stv = !!STV,
                     ry = !!RY,
                     #sma = sma,
                     target = target,
                     confidence = confidence) %>%
    # AC: With this line, in the summary tabla always printed remove2x = "TRUE",
      #It should print the chosen option instead
        mutate(remove.leverage = remove.leverage,
    # AC: I added a line to count the number of observation used for the CSTV estimations
               n = nrow(.))
        #mutate(remove2x = "TRUE")

    if(summary == TRUE) {
        return(
            stage_3 %>%
                select(
                    #model,
                    n, # new outcome
                    target,
                    cstv,
                    lower_cstv,
                    upper_cstv,
                    confidence,
                    pvalue,
                    r,
                    # Replaced remove2x
                    remove.leverage) %>%
                distinct(across(everything()))
        )
    } else {
        return(stage_3)
    }

}
```

6.1. Plot function

```r
# Function alcc_plot() creates a scatter plot of soil test correlation data
# arguments are passed to alcc()
alcc_plot.rev <- function(data,
                      stv,
                      ry,
                      #sma = TRUE,
                      target,
                      #sufficiency = 90,
                      confidence,
                      remove.leverage = "none") {

    if (missing(stv)) {
        stop("Specify the soil test variable (e.g., stv = STK)")
    }

    if (missing(ry)) {
        stop("Enter name of relative yield (%) variable (e.g., ry = RY)")
    }

    # AC: # Add statement for missing target
    if (missing(target)) {
        stop("Specify the target value of relative yield (%) (e.g., target = 90)")
    }

    # AC: # Add statement for missing confidence
    if (missing(confidence)) {
        stop("Specify the desired confidence level (%) for cstv estimation (e.g., confidence = 95)")
    }

    # enquo let's the user enter their own column names for x and y variables
    # AC: Replaced names
    STV <- enquo(stv)
    RY <- enquo(ry)
    #x <- enquo(stv)
    #y <- enquo(ry)


    input <- data %>%
        transmute(stv = !!STV,
                  # RY values greater than 100 are capped at 100
# AC: the false statement should always be of type double, just in case,
# use as.double()
# I have just tried with a dataset where I found this problem (data_2)
                  ry_cap = if_else(!!RY > 100, 100, as.double(!!RY)))

    # pass to alcc function
    output <- alcc.rev(data,
                   stv = !!STV,
                   ry = !!RY,
                   # AC: sma not needed
                   # sma = sma,
                   target =  target,
                   confidence = confidence,
              # AC: I changed remove2x for remove.leverage
                   remove.leverage = remove.leverage)

    # for plot annotations

    maxx <- max(input$stv)

    cstv <- (unique(output$cstv))
    cstv <- if_else(cstv < 10, round(cstv, 1), round(cstv, 0))

    lower_cstv <- (unique(output$lower_cstv))
    lower_cstv <- if_else(lower_cstv < 10, round(lower_cstv, 1), round(lower_cstv, 0))

    upper_cstv <- (unique(output$upper_cstv))
    upper_cstv <- if_else(upper_cstv < 10, round(upper_cstv, 1), round(upper_cstv, 0))

    #sufficiency <- unique(output$sufficiency)

    r <- round(unique(output$r), 2)

# AC: WARNING! WHAT IF I DON'T WANT TO REMOVE ANYTHING? Currently, the default
  # is to remove STV > CSTV100
# Default option should be using all available data, then the remove2x = TRUE/FALSE
# plot includes all data, even if remove2x == TRUE, but curve will follow remove2x
    output %>%
        ggplot() +
        geom_point(data = input,
                   aes(stv, ry_cap),
                   size = 2, alpha = 0.7,
                   color = case_when(
                              # AC: I changed the remove2x arg. for remove.leverage
                              # None
                              remove.leverage == "none" & input$stv <= cstv_100 ~ "black",
                              remove.leverage == "none" & input$stv > cstv_100 ~ "purple",
                              # >100
                              remove.leverage == "cstv100" & input$stv <= cstv_100 ~ "black",
                              remove.leverage == "cstv100" & input$stv > cstv_100 ~ 'red',
                              # >90_2x
                              remove.leverage == "cstv90_2x" & input$stv <= cstv90_2x ~ "black",
                              remove.leverage == "cstv90_2x" & input$stv > cstv90_2x ~ "red"
                              #remove.leverage == TRUE & input$stv > cstv90_2x ~ "red"
                              #remove2x == FALSE & input$stv > cstv_100 ~ red,
                              #remove.leverage == FALSE & input$stv > cstv90_2x ~ gold,
                              #remove.leverage == TRUE & input$stv > cstv90_2x ~ red,
                              #TRUE ~ black
                              )) +
        geom_vline(xintercept = lower_cstv,
                   alpha = 0.5,
                   linetype = 3) +
        geom_vline(xintercept = upper_cstv,
                   alpha = 0.5,
                   linetype = 3) +
        geom_line(aes(x = fitted_stv,
                      y = fitted_ry),
                  color = "red", size = 1.5) +
        # the cstv at X% sufficiency
        geom_vline(xintercept = cstv, color = "blue", alpha = 1) +
        geom_hline(yintercept = target, alpha = 0.2) +
        annotate(
            "text",
            label = paste0(
                "CSTV = ", cstv, " ppm at ", target, "% RY",
                "\n", confidence,"% CI [", lower_cstv, " - ", upper_cstv, "]",
                "\n r = ", r),
            x = maxx,
            y = 0,
            hjust = 1,
            vjust = 0,
            alpha = 0.8
        ) +
        scale_x_continuous(breaks =
                               seq(0, 10000, by = if_else(
                                   maxx >= 200, 40,
                                   if_else(maxx >= 100, 20,
                                           if_else(maxx >= 50, 10, 5))
                               ))) +
        scale_y_continuous(breaks = seq(0, 105, 10)) +
        labs(subtitle = "Modified Arcsine-Log Calibration Curve",
  # AC: Here there could be a problem for people using STV in other than mg/kg ...
            x = expression(Soil ~ Test ~ Value ~ (mg ~ kg ^ -1)),
            y = "Relative yield (%)",
            caption = paste0(
            # AC: I Replaced this in line with the new arg. remove.leverage
  #if_else(remove.leverage == TRUE,
  #"Red points > CSTV90 * 2 excluded from model",
  #"Red points > CSTV100 excluded from model. Yellow points > CSTV90 * 2 not excluded."),
  case_when(remove.leverage == "none" ~ "No points excluded from model. Purple points (>CSTV100)",
            remove.leverage == "cstv100" ~ "Red points (STV > CSTV100) were excluded from model",
            remove.leverage == "cstv90_2x" ~ "Red points (STV > 2*CSTV90) were excluded from model"),
                "\nVertical dotted lines show lower and upper confidence limits"
            )
        )
}
```

7. Comparison to SoilTestCocaCola

7.1. Cotton (from agridat package)

For the cotton dataset, Austin's Code using alcc() produces:

i. different cstv, lower & upper limits when compared to Adrian's ALCC() and the reference (SMA regression). This issue is nothing wrong with formulae but a result of using a dataset that presents points with STV > cstv100, since the alcc() removes those points by default, while Adrian's ALCC() does not, as I prefer to leave this decision up to the user.

ii. wrong p=values, since it uses the student-t one instead of the pvalue to test correlation significance (used by SMA regression).

```
# Load  Austin's ALCC functions
# alcc.core() and alcc()
source_url("https://raw.githubusercontent.com/austinwpearce/SoilTestCocaCola/main/alcc.R")
# alcc_plot()
#source_url("https://raw.githubusercontent.com/austinwpearce/SoilTestCocaCola/main/alcc_plot.R")

# EXAMPLE 1. Cotton. Relative yield vs soil test potassium
cotton <- tibble(stk = agridat::cate.potassium$potassium,
                 ry = agridat::cate.potassium$yield,
                 dataset = "cotton")

### CODES COMPARISON

# Adrian's
ALCC(data = cotton, RY = ry, STV = stk, target=90, confidence = 0.95)
```

```
## Warning: One or more original RY values exceeded 100%.
## All RY values greater than 100% have been capped to 100% for arcsine transformation.
```

```
## # A tibble: 1 x 9
##       n     r   p_value target confidence  CSTV    LL    UL Curve
##   <int> <dbl>     <dbl>  <dbl>      <dbl> <dbl> <dbl> <dbl> <list>
## 1    24 0.728 0.0000557     90       0.95  76.9  62.3  94.8 <tibble [297 x 2]>
```

```
# Austin's, sma = TRUE
# Check that even stating remove2x = FALSE it always print "remove2x = TRUE" on the summary table.
alcc(cotton, soil_test = stk, ry = ry, sufficiency = 90,
     confidence = 95, summary = T, remove2x = FALSE, sma = T)
```

```
## Warning: One or more original RY values exceeded 100%.
## All RY values greater than 100% have been capped to 100% for arcsine transformation.
```

```
## # A tibble: 1 x 9
##   model  sufficiency  cstv lower_cl upper_cl confidence  pvalue pearson remove2x
##   <chr>        <dbl> <dbl>    <dbl>    <dbl>      <dbl>   <dbl>   <dbl> <chr>
## 1 ALCC-~          90  73.5     59.2     91.2         95 1.54e-4   0.694 TRUE
```

```
# Austin's, sma = FALSE
alcc(cotton, soil_test = stk, ry = ry, sufficiency = 90,
     confidence = 95, summary = T, remove2x = FALSE, sma = F)
```

```
## Warning: One or more original RY values exceeded 100%.
## All RY values greater than 100% have been capped to 100% for arcsine transformation.

## # A tibble: 1 x 9
##   model sufficiency  cstv lower_cl upper_cl confidence   pvalue pearson remove2x
##   <chr>       <dbl> <dbl>    <dbl>    <dbl>      <dbl>    <dbl>   <dbl> <chr>
## 1 ALCC           90  67.5     55.9     81.5         95  8.89e-5   0.725 TRUE
```

```
## For the Austin's Code using alcc() ->
# cstv, lower & upper limit, and p=values are wrong compared to
# SMA regression as reference.
# Austin's results (REFERENCE) [ADRIAN's]:
  # CSTV = 67.49 (76.89) [76.89]
  # LL = 55.87 (62.36) [62.35]
  # UL = 81.54 (94.79) [94.83]
  # pval = 8.87e-05 (5.57e-05) [5.57e-05]

# NOW COMPARE TO MY REVISED VERSION of alcc (alcc.rev)

# Adrian's
ALCC(data = cotton, RY = ry, STV = stk, target=90, confidence = 0.95)
```

```
## Warning: One or more original RY values exceeded 100%.
## All RY values greater than 100% have been capped to 100% for arcsine transformation.

## # A tibble: 1 x 9
##       n     r   p_value target confidence  CSTV    LL    UL Curve
##   <int> <dbl>     <dbl>  <dbl>      <dbl> <dbl> <dbl> <dbl> <list>
## 1    24 0.728 0.0000557     90       0.95  76.9  62.3  94.8 <tibble [297 x 2]>
```

```
# Revised alcc functions
alcc.rev(cotton, stv = stk, ry = ry, target = 90,
     confidence = 95, summary = T, remove.leverage = "none")
```

```
## Warning: One or more original RY values exceeded 100%.
## All RY values greater than 100% have been capped to 100% for arcsine transformation.

## # A tibble: 1 x 9
##       n target  cstv lower_cstv upper_cstv confidence    pvalue     r
##   <int>  <dbl> <dbl>      <dbl>      <dbl>      <dbl>     <dbl> <dbl>
## 1    24     90  76.9       62.3       94.8         95 0.0000557 0.728
## # ... with 1 more variable: remove.leverage <chr>
```

```
alcc.rev(cotton, stv = stk, ry = ry, target = 90,
     confidence = 95, summary = T, remove.leverage = "cstv100")
```

```
## Warning: One or more original RY values exceeded 100%.
## All RY values greater than 100% have been capped to 100% for arcsine transformation.
```

```
## # A tibble: 1 x 9
##       n target  cstv lower_cstv upper_cstv confidence   pvalue     r
##   <int>  <dbl> <dbl>      <dbl>      <dbl>      <dbl>    <dbl> <dbl>
## 1    22     90  73.5       59.2       91.2         95 0.000339 0.694
## # ... with 1 more variable: remove.leverage <chr>
```

```
alcc.rev(cotton, stv = stk, ry = ry, target = 90,
     confidence = 95, summary = T, remove.leverage = "cstv90_2x")
```

```
## Warning: One or more original RY values exceeded 100%.
## All RY values greater than 100% have been capped to 100% for arcsine transformation.
```

```
## # A tibble: 1 x 9
##       n target  cstv lower_cstv upper_cstv confidence   pvalue     r
##   <int>  <dbl> <dbl>      <dbl>      <dbl>      <dbl>    <dbl> <dbl>
## 1    22     90  73.5       59.2       91.2         95 0.000339 0.694
## # ... with 1 more variable: remove.leverage <chr>
```

7.1.2. Double-check with SMA (reference method)

```r
# USE SMA as reference method

target = 90
cotton <- cotton %>% mutate(# Transform stv
                            ln_stv = log(stk),
                            # Cap if RY > 100
                            RY = ifelse(ry > 100, 100, as.double(ry)),
                            # Transform and center RY
                            arc_ry = (asin(sqrt(RY/100)) - asin(sqrt(target/100))))

# SMA model
sma.cotton <- sma(data = cotton, method = c("SMA"), formula = ln_stv ~  arc_ry)

# Print
sma.cotton
```

```
## Call: sma(formula = ln_stv ~ arc_ry, data = cotton, method = c("SMA"))
##
## Fit using Standardized Major Axis
##
## ------------------------------------------------------------
## Coefficients:
##             elevation    slope
## estimate     4.342399 2.479801
## lower limit  4.133033 1.839335
## upper limit  4.551764 3.343281
##
## H0 : variables uncorrelated
## R-squared : 0.529641
## P-value : 5.5672e-05
```

```r
# Extract intercept, its CI, and model p-value
print("CSTV from SMA")
```

```
## [1] "CSTV from SMA"
```

```r
exp(sma.cotton[["elevtest"]][[1]][["a"]])
```

```
## [1] 76.89175
```

```r
print("Lower limit")
```

```
## [1] "Lower limit"
```

```r
exp(sma.cotton[["elevtest"]][[1]][["a.ci"]][[1]])
```

```
## [1] 62.36682
```

```r
print("Upper limit")
```

```
## [1] "Upper limit"
```

```r
exp(sma.cotton[["elevtest"]][[1]][["a.ci"]][[2]])
```

```
## [1] 94.79947
```

```r
print("p-value")
```

```
## [1] "p-value"
```

```r
sma.cotton[["pval"]][[1]]
```

```
## [1] 5.567174e-05
```

7.2. Other examples

Data_1: The alcc() from Austin works the same than Adrian's ALCC() and Spreadsheet version.

Data_2: Austin's original alcc() code fails with data_2 due to an issue with the "if_else()" used to cap RY values. The function needs to have specified the FALSE statement as.double(), otherwise it may result problematic with some datasets loaded from csv files (as data_2).

```
# Adrian's
fit_example_1 = ALCC(data = data_1, RY = RY, STV = STV, target=90, confidence = 0.95)
fit_example_1
```

```
## # A tibble: 1 x 9
##       n     r       p_value target confidence  CSTV    LL    UL Curve
##   <int> <dbl>         <dbl>  <dbl>      <dbl> <dbl> <dbl> <dbl> <list>
## 1    15 0.968 0.00000000330     90       0.95  4.48  3.95  5.08 <tibble>
```

```
# Austin's
alcc(data_1, soil_test = STV, ry = RY, sufficiency = 90,
     confidence = 95, summary = T, remove2x = FALSE, sma = TRUE)
```

```
##      model sufficiency     cstv lower_cl upper_cl confidence       pvalue
## 1 ALCC-SMA          90 4.478476 3.947041 5.081463         95 6.469298e-10
##     pearson remove2x
## 1 0.9682908     TRUE
```

```
# Fixed alcc.rev
alcc.rev(data_1, stv = STV, ry = RY, target = 90,
     confidence = 95, summary = T, remove.leverage = "none")
```

```
##    n target     cstv lower_cstv upper_cstv confidence       pvalue         r
## 1 15     90 4.478476   3.947041   5.081463         95 3.296044e-09 0.9682908
##   remove.leverage
## 1            none
```

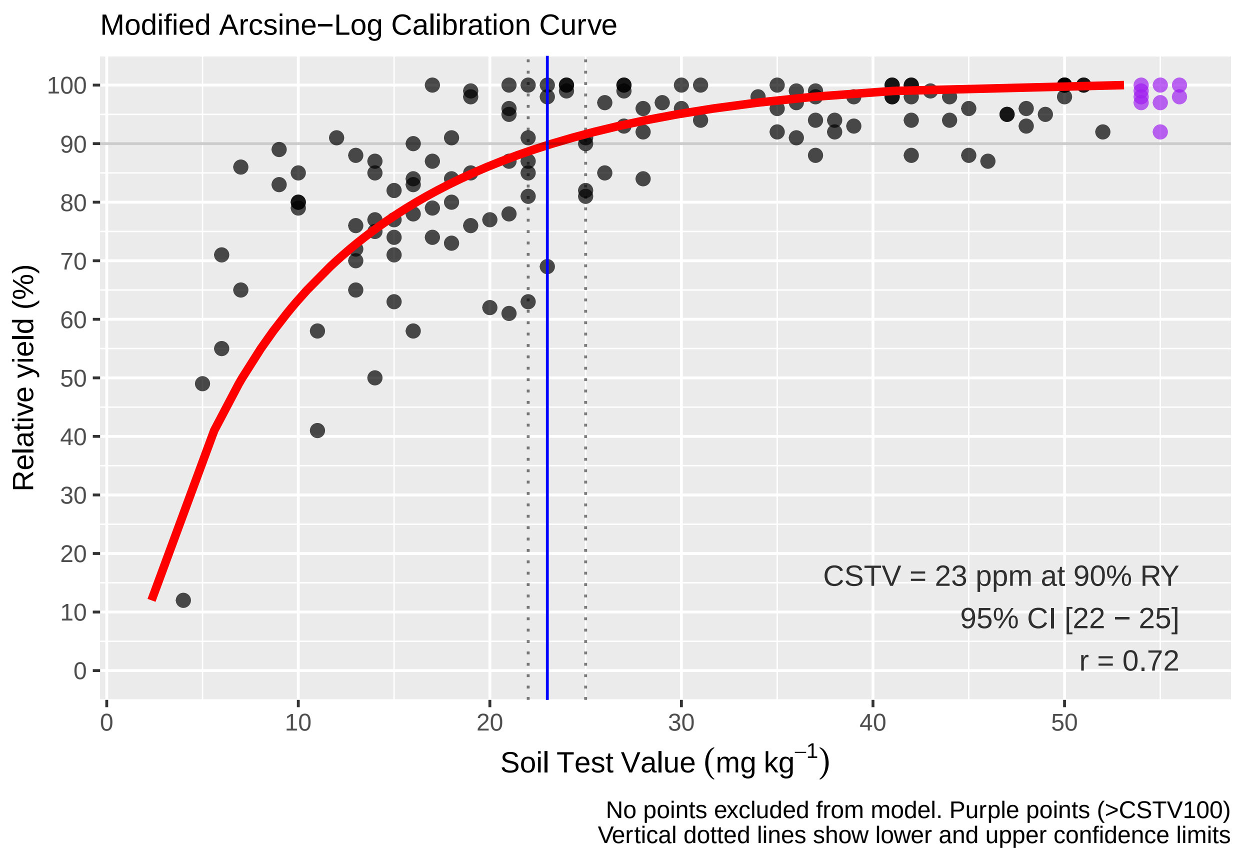
```
# Using dataframe

# Alternative using the vectors
# fit_example_1 = ALCC(RY = data_1$RY,STV = data_1$STV, target=90,confidence = 0.95)

# Adrian's
fit_example_2 = ALCC(data = data_2, RY = RY, STV = STV, target=90, confidence = 0.95)
fit_example_2
```

```
## # A tibble: 1 x 9
##       n     r  p_value target confidence  CSTV    LL    UL Curve
##   <int> <dbl>    <dbl>  <dbl>      <dbl> <dbl> <dbl> <dbl> <list>
## 1   137 0.716 7.31e-23     90       0.95  23.3  21.6  25.1 <tibble [441 x 2]>
```

```
# Revised alcc.rev
#
# remove.leverage = "none"
alcc.rev(data_2, stv = STV, ry = RY, target = 90,
     confidence = 95, summary = T, remove.leverage = "none")
```

```
##     n target     cstv lower_cstv upper_cstv confidence       pvalue         r
## 1 137     90 23.25457   21.57156   25.06888         95 7.314913e-23 0.7164928
##   remove.leverage
## 1            none
```

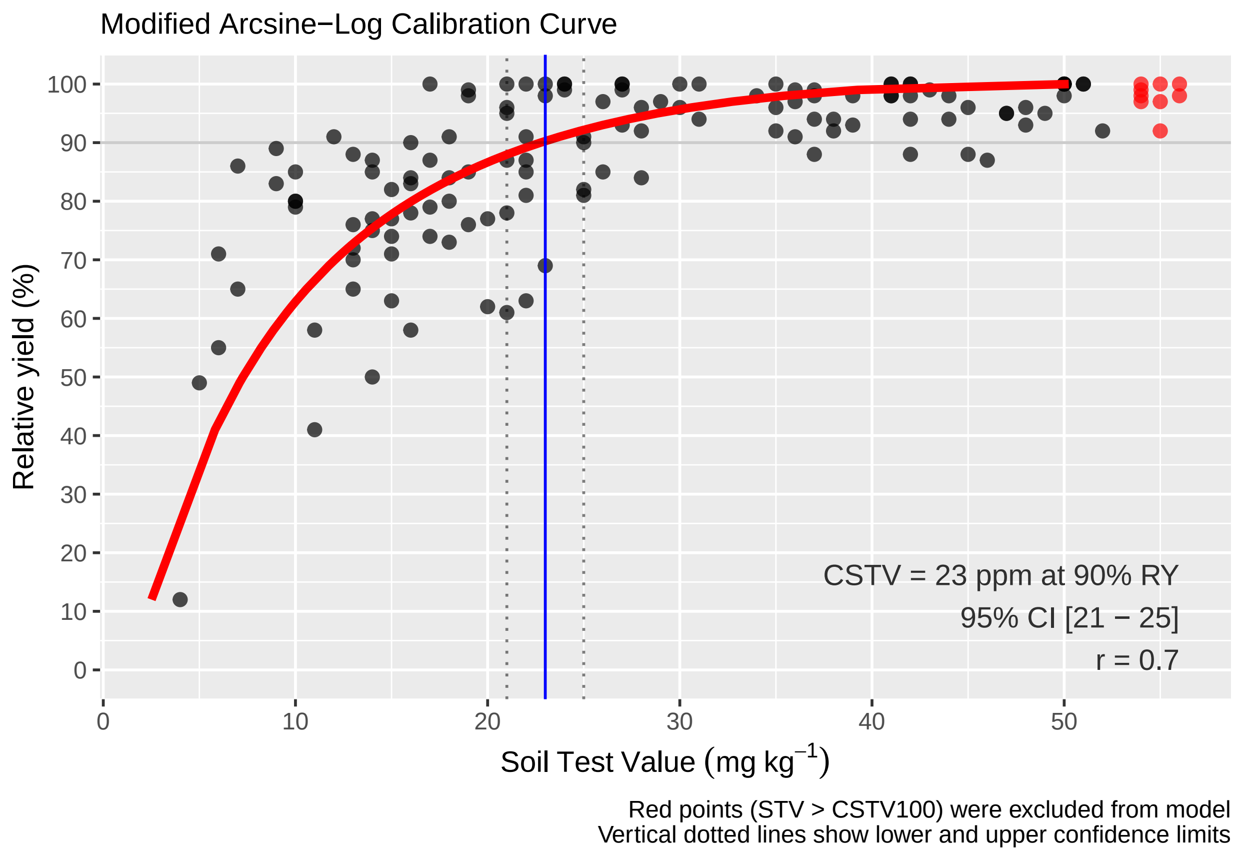
```
# plot
alcc_plot.rev(data_2, stv = STV, ry = RY, target = 90,
     confidence = 95, remove.leverage = "none")
```



```
# remove.leverage = "cstv100"
alcc.rev(data_2, stv = STV, ry = RY, target = 90,
     confidence = 95, summary = T, remove.leverage = "cstv100")
```

```
##     n target     cstv lower_cstv upper_cstv confidence       pvalue         r
## 1 128     90 22.74894   21.07941    24.5507         95 1.722875e-20 0.7044885
##   remove.leverage
## 1         cstv100
```

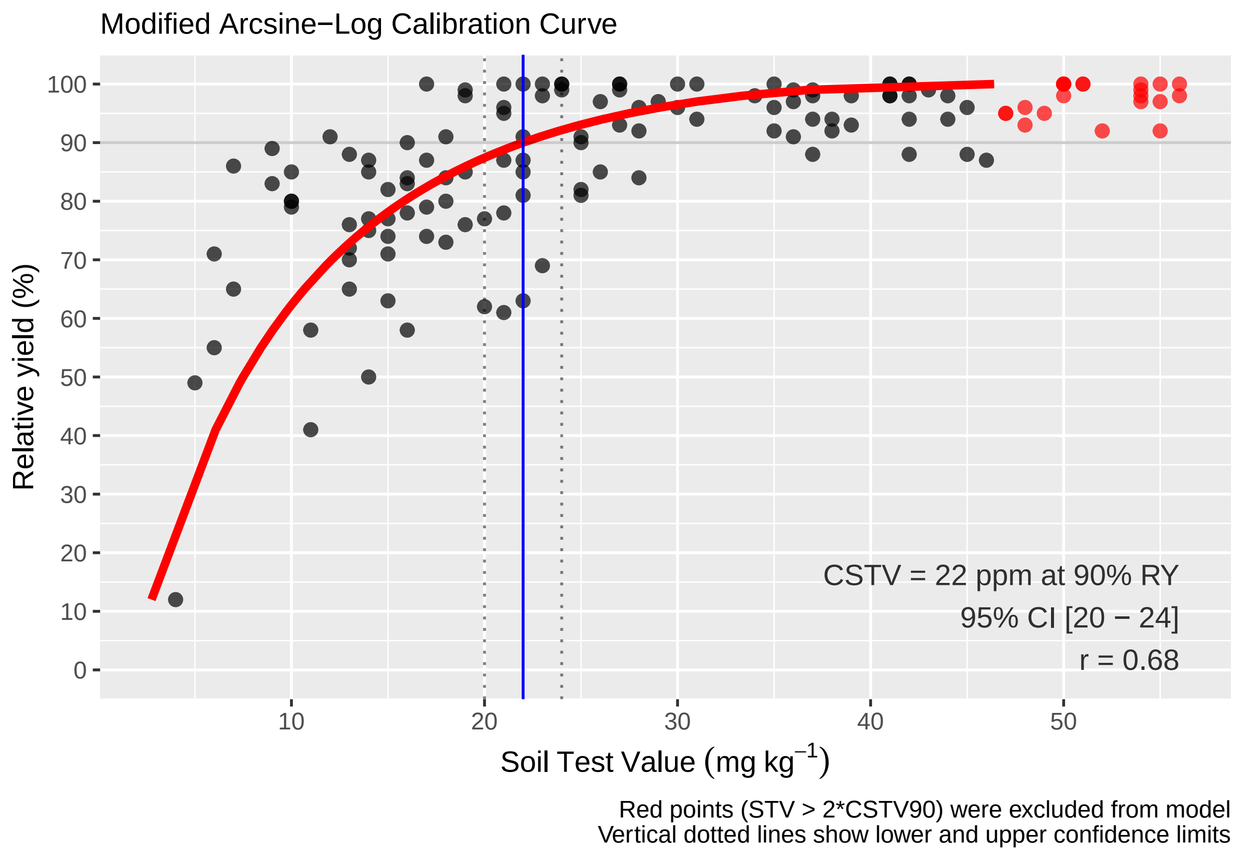
```
# plot
alcc_plot.rev(data_2, stv = STV, ry = RY, target = 90,
     confidence = 95, remove.leverage = "cstv100")
```



```r
# remove.leverage = "cstv90_2x"
alcc.rev(data_2, stv = STV, ry = RY, target = 90,
     confidence = 95, summary = T, remove.leverage = "cstv90_2x")
```

```
##     n target     cstv lower_cstv upper_cstv confidence       pvalue         r
## 1 115     90 21.98504   20.32021   23.78626         95 5.322914e-17 0.6813117
##   remove.leverage
## 1       cstv90_2x
```

```r
# plot
alcc_plot.rev(data_2, stv = STV, ry = RY, target = 90,
     confidence = 95, remove.leverage = "cstv90_2x")
```



```
# Austin's original alcc() fails with data_2
# Austin's code fails due to issue with the if_else function to cap RY values.
# For some datasets loaded from csv files, the function needs to have specified the FALSE statement as.
# alcc(data_2, soil_test = STV, ry = RY, sufficiency = 90, confidence = 95, summary = T, remove2x = FAL
```

7.3. Extra example data_3

Data_3: Austin's alcc() does not produce the same results than the ALCC() or the Spreadsheet version. Again, this is related to the fact that Austin's alcc() always remove points STV > CSTV100 when present, so it produces different results than Adrian's ALCC().

Using the Adrian's revised alcc.rev() functions, Austin's alcc() is equivalent to use the alcc.rev(remove.leverage = 'cstv100').

```
# Example 3 dataframe. Imported from xlsx file
data_3 = readxl::read_xlsx(path ="data_3.xlsx", sheet = 1)

# Adrian's ALCC()
ALCC(data = data_3, RY = RY, STV = STV, target=90, confidence = 0.95)
```

```
## # A tibble: 1 x 9
##       n     r   p_value target confidence  CSTV    LL    UL Curve
##   <int> <dbl>     <dbl>  <dbl>      <dbl> <dbl> <dbl> <dbl> <list>
## 1   107 0.374 0.0000741     90       0.95  21.8  19.4  24.5 <tibble [376 x 2]>
```

```
# Austin's alcc()
alcc(data_3, soil_test = STV, ry = RY, sufficiency = 90,
     confidence = 95, summary = T, remove2x = FALSE, sma = TRUE)
```

```
## # A tibble: 1 x 9
##   model  sufficiency  cstv lower_cl upper_cl confidence  pvalue pearson remove2x
##   <chr>        <dbl> <dbl>    <dbl>    <dbl>      <dbl>   <dbl>   <dbl> <chr>
## 1 ALCC-~          90  21.1     18.7     23.7         95 0.00136   0.291 TRUE
```

```
# Fixed alcc.rev(remove.leverage = "none")
alcc.rev(data_3, stv = STV, ry = RY, target = 90,
     confidence = 95, summary = T, remove.leverage = "none")
```

```
## # A tibble: 1 x 9
##       n target  cstv lower_cstv upper_cstv confidence    pvalue     r
##   <int>  <dbl> <dbl>      <dbl>      <dbl>      <dbl>     <dbl> <dbl>
## 1   107     90  21.8       19.4       24.5         95 0.0000741 0.374
## # ... with 1 more variable: remove.leverage <chr>
```

```
# Fixed alcc.rev(remove.leverage = "cstv100")
alcc.rev(data_3, stv = STV, ry = RY, target = 90,
     confidence = 95, summary = T, remove.leverage = "cstv100")
```

```
## # A tibble: 1 x 9
##       n target  cstv lower_cstv upper_cstv confidence  pvalue     r
##   <int>  <dbl> <dbl>      <dbl>      <dbl>      <dbl>   <dbl> <dbl>
## 1   104     90  21.1       18.7       23.7         95 0.00272 0.291
## # ... with 1 more variable: remove.leverage <chr>
```

```
# Fixed alcc.rev(remove.leverage = "cstv90_2x")
alcc.rev(data_3, stv = STV, ry = RY, target = 90,
     confidence = 95, summary = T, remove.leverage = "cstv90_2x")
```

```
## # A tibble: 1 x 9
##       n target  cstv lower_cstv upper_cstv confidence  pvalue     r
##   <int>  <dbl> <dbl>      <dbl>      <dbl>      <dbl>   <dbl> <dbl>
## 1   104     90  21.1       18.7       23.7         95 0.00272 0.291
## # ... with 1 more variable: remove.leverage <chr>
```