Analysis of Real-World Gas-Exchange Data using Gasanalyzer

```
library(gasanalyzer)
library(units)
```

```
library(ggplot2)
library(gridExtra)
```

GasanalyzeR

This R package provides methods to read and process data from different portable photosynthesis systems. Variable names used in vendor-specific data files are translated into uniform names. Metadata, such as the instrument name and filename, are also added. For files that do not store timezone information, the local timezone is used by default.

Files from different instruments can be read. For the LI-COR 6800 gas-analyzer, both text files and the XML-based `.xlsx` files can be imported. In case `.xlsx` files are used, the equations stored in the files are converted to `R` and can be used to recalculate the data.

The three examples discussed below demonstrate the use of the package.

1. LI-6800 data and recalculations

The data used in this example is a CO_2 response curve measured at low oxygen. Before loading the data files, the calibration factors for the instrument used to acquire the data must be imported. This is only necessary for some advanced functionality used later, and currently only available for calibration factors stored by the LI-6800 instrument.

The code below shows how to load the data from the `.txt` and `.xlsx` files.

```
exampleFiles <- system.file("extdata", package = "gasanalyzer")
# import calibration factors for the 6800 used here:
import_factory_cals(exampleFiles)
# load data
xlsxfile <- read_6800_xlsx(paste(exampleFiles, "lowo2.xlsx", sep = "/"))
txtfile <- read_6800_txt(paste(exampleFiles, "lowo2", sep = "/"))
```

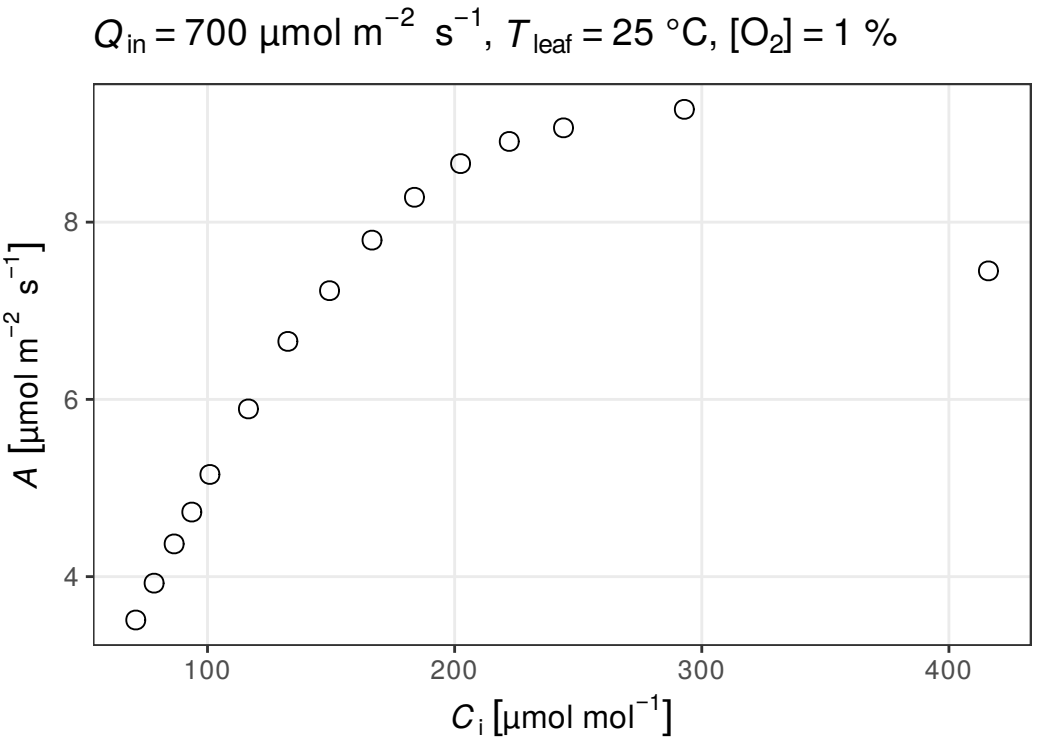
The LI-6800 confusingly stores the results of divisions by zero as zero in the `txt` file. After excluding columns containing the results of divisions by zero and the column with equations (a list column), there are very few differences between both files:

```
# exclude list columns with equations, and divisions by zero
excludeCols <- vapply(xlsxfile,
                      FUN=\(x) is.list(x) ||
                        all(is.infinite(x)) ||
                        all(is.nan(x)), TRUE)

# Interestingly, Leak.Fan reported in the txt file differs slightly
# from that in the xlsx. It is likely caused by rounding/averaging in the
# instrument firmware.
```

```
all.equal(txtfile[!excludeCols], xlsxfile[!excludeCols], tolerance = 1e-6)
#> [1] "Component \"Leak.Fan\": Mean relative difference: 0.0012"
```

Using plotting facilities available with R, relationships in the data can be visualized. In the following example, `var2label()` was used to replace the variable names in the data with commonly-used symbols for the plot. The relation between net photosynthesis (A) and intercellular CO_2 (C_i) is shown.

```
# default points:
update_geom_defaults("point", list(shape = 1, size = 3))

# convenience function to change plot labels automatically:
nice_labels <- function(plt) {
  plt$labels <- var2label(plt$labels)
  plt
}

#generate some descriptive stats  and plotmath it with a call to var2label:
descr <- colMeans(xlsxfile[c("LeafQ.Qin", "Meas.Tleaf", "Const.Oxygen")])
dlist <- var2label(rep(names(descr), 2),
                   use_units = rep(c(FALSE, TRUE), each = 3),
                   val = c(rep(NA, 3), as.character(descr)),
                   group = c("", "")) |>
    setNames(letters[1:6]) |> unlist() |>
    substitute(a==d*","~b==e*","~c==f, env = _ ) |> as.expression()

AvsCi1 <- xlsxfile |> ggplot(aes(GasEx.Ci, GasEx.A)) + geom_point() +
  labs(title = dlist)

nice_labels(AvsCi1)
#> Warning: The `scale_name` argument of `continuous_scale()` is deprecated as of ggplot2
#> 3.5.0.
#> This warning is displayed once every 8 hours.
#> Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
#> generated.
```



If a researcher would like to make changes to some of constants used by the instrument for calculating derived variables, all data should be recalculated to take the effect into account. For example, an application note by the instrument vendor[1] shows how to use complimentary measurements with a porometer to determine the ratio of stomatal conductances between both sides of the leaves. For the leaf in this experiment, the ratio was 0.22. Such corrections typically have only very small effects because of the high boundary layer conductance in the leaf chambers that are commonly used for such experiments.

```
gswOld <- xlsxfile$GasEx.gsw
xlsxfile <- xlsxfile |>
  transform(Const.K = 0.22) |>
  recalculate()
#max effect:
set_units(max(1 - as.numeric(xlsxfile$GasEx.gsw / gswOld)) * 100, "%")
#> -0.76 [%]
```

Gas-exchange measurements can be combined with measurements of chlorophyll fluorescence to obtain estimates of the electron transport rate (J_F) through photosystem 2 (PSII). However, this requires knowledge of the amount of light energy absorbed by the leaf (α), and the fraction of energy partitioned to PSII (β[2]):

$$J_F = \alpha\beta\Phi_{PSII}Q_{in}$$

where Φ_{PSII} is the operating quantum efficiency of PSII, and Q_{in} is the light intensity incident on the leaf.

Calculations implemented by the vendor typically assume β is 0.5 and α is around 0.85 (possibly adjusted for the spectrum of the light source used). An interesting application of this electron transport estimate is to to back-calculate the CO_2 concentration at the site of carboxylation, and by extension estimate the mesophyll

[1] LI-COR Biosciences Inc (2022) Photosynthesis | Complementary Leaf Physiology Measurements: The LI-600 and LI-6800. Link

[2] Genty B, Briantais J-M, Baker NR (1989) The relationship between the quantum yield of photosynthetic electron transport and quenching of chlorophyll fluorescence. Biochimica et Biophysica Acta (BBA) - General Subjects 990:87–92. Link

conductance (g_m[3]).

However, leaves vary in the amount of light absorbed. Moreover, not all of the absorbed light is used for photosynthetic carbon reduction. To address such concerns, an additional estimate for the electron transport rate (J_C) may be obtained by measuring gas exchange under non-photorespiratory conditions (i.e. at low oxygen). In this case, it follows from the C_3 biochemical model[4] that:

$$J_C \approx 4(A + R_L) \rightarrow A \approx J_C/4 - R_L$$

where R_L represents the dark respiration rate in the light. Assuming that alternative electron flows are minimal, the following relationship can be derived (Yin et al. 2009[5]):

$$A \approx J/4 - R_L = \alpha\beta\Phi_{PSII}Q_{in}/4 - R_L$$

However, applying this assumptions on the data used here shows that either this model is too simple, or the assumed values for α and/or β are incorrect:

```
#confirm that the slope of A vs ETR is not 1:
xlsxfile |>
    lm(GasEx.A ~ I(FLR.ETR/4), data = _) |> coef() |> _[[2]]
#> [1] 1.1
```

If the goal is to estimate g_m, it has been recommended (e.g. Yin et al. 2009) to not rely on the $\alpha\beta$ outputted by the instrument (0.42 in this case), but instead use a linear regression to find an "effective" value:

```
# a linear regression:
lm1 <-
  xlsxfile |> transform(FLR.PhiQin_4 = FLR.phiPS2 * LeafQ.Qin / 4) |>
  lm(GasEx.A ~ FLR.PhiQin_4, data = _)

alphabeta <- lm1$coef[[2]]
```

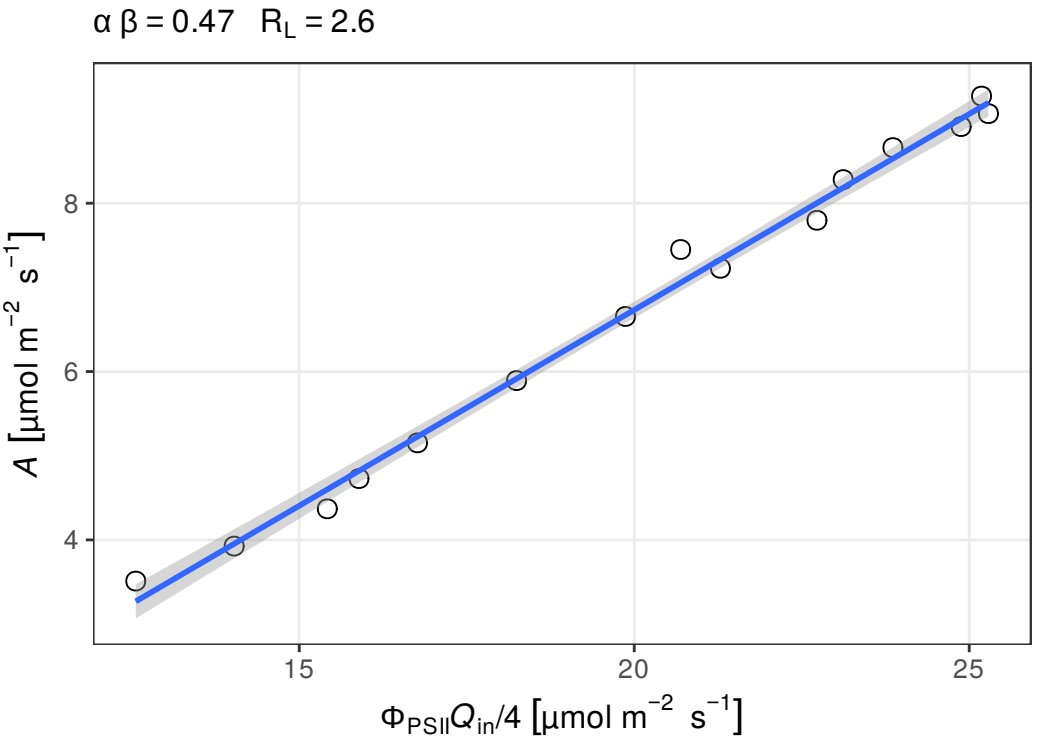
The following plots the result of the regression:

```
AvsPhiQin1 <- ggplot(lm1$model, aes(x = .data[[names(lm1$model)[2]]],
                    y = .data[[names(lm1$model)[1]]])) +
    geom_point() +
    stat_smooth(method = "lm", formula = y ~ x) +
    labs(subtitle = bquote(alpha ~ beta == .(lm1$coef[[2]]) ~~~
                               R[L] == .(-lm1$coef[[1]])))

nice_labels(AvsPhiQin1)
```

[3]Harley PC, Loreto F, Di Marco G, Sharkey TD (1992) Theoretical Considerations when Estimating the Mesophyll Conductance to CO_2 Flux by Analysis of the Response of Photosynthesis to CO_2. Plant Physiol 98:1429–1436. Link

[4]Farquhar GD, von Caemmerer S, Berry JA (1980) A biochemical model of photosynthetic CO2 assimilation in leaves of C3 species. Planta 149:78–90. Link

[5]Yin X, van der Putten PEL, Belay D, Struik PC (2020) Using photorespiratory oxygen response to analyse leaf mesophyll resistance. Photosynth Res 144:85–99. Link



The fitted value for the slope can now be used to adjust the constant value used by the instrument. The default equations used by the LI-6800 recalculate α based on the spectral composition of the light source. These equations are stored as `R` calls in the `gasanalyzer.Equations` column. They can also be manually extracted from a given `xlsx` file using `extract_6800_equations()` and edited using `modify_equations()`:

```
# a long equation estimating the light absorption:
xlsxEqs <- xlsxfile$gasanalyzer.Equations[[1]]
print(xlsxEqs$LeafQ.alpha)
#> (QConst.fQambIn * LQConst.AbsAmbient + QConst.fQambOut * LQConst.AbsAmbient +
#>     QConst.fQflr * ((FlrLS.Qred + FlrLS.Qmodavg)/pmax(FlrLS.Qred +
#>         FlrLS.Qmodavg + FlrLS.Qblue, 0.1) * LQConst.RedAbsFlr +
#>         FlrLS.Qblue/pmax(FlrLS.Qred + FlrLS.Qmodavg + FlrLS.Qblue,
#>             0.1) * LQConst.BlueAbsFlr))/(QConst.fQambIn + QConst.fQambOut +
#>     QConst.fQflr)
```

```
# add our custom alpha by dividing it through the used beta:
xlsxfile$Custom.alpha <- alphabeta / xlsxfile$Const.fPS2
# and change the equation to use the new value:
xlsxEqs <- modify_equations(xlsxEqs, LeafQ.alpha = \() {Custom.alpha})

# make a copy of the data and apply the new equations:
xlsxfileCopy <- xlsxfile |>
  transform(gasanalyzer.Equations = list(xlsxEqs)) |>
  recalculate()

# confirm that the slope of A vs ETR is now 1:
xlsxfileCopy |>
  lm(GasEx.A ~ I(FLR.ETR / 4), data = _) |> coef() |> _[[2]]
#> [1] 1
```

The `xlsx` equations imported from the file are typically sufficient for simple recalculations, but limited to

the specific set of equations defined in the file. Moreover, in case only text-based data files are available, an alternative way to recalculate data is required. To handle calculations more comprehensively for all types of data, `gasanalyzer` includes a several sets of equations related to gas exchange, chlorophyll fluorescence and carbon isotope discrimination. To ensure greater reliability, units of the used quantities in these equations are automatically enforced by the `units` package[6]. In the code below, the data is recalculated using the default set, with additional instrument specific computations for the LI-6800. In addition, a several equations are added to take the oxygen concentration stored in the data into account when recalculating the gas mole fractions. Finally, α is set to the previously fitted value.

```
RecalEqs <- create_equations(c("li6800", "default", "O2_correction"),
                             LeafQ.alpha = \() {Custom.alpha})
xlsxfileRecal <- recalculate(xlsxfile, RecalEqs)

all.equal(xlsxfileRecal[names(xlsxfile)], xlsxfile, tol = 0.001)
#> [1] "Component \"GasEx.NetTherm\": Mean relative difference: 0.0015"
#> [2] "Component \"LeafQ.Qabs\": Mean relative difference: 0.096"
#> [3] "Component \"LeafQ.alpha\": Mean relative difference: 0.096"
#> [4] "Component \"FLR.QabsFs\": Mean relative difference: 0.096"
#> [5] "Component \"FLR.ETR\": Mean relative difference: 0.096"
#> [6] "Component \"FLR.phiCO2\": Mean relative difference: 0.11"
#> [7] "Component \"gasanalyzer.UseEqUnits\": 15 element mismatches"
```

Small differences are found in variables as a result of the new α and in a value related to the energy balance. The latter is a result of a more precise conversion from Celsius to Kelvin units. Equations can be checked and modified before use, and custom equations can be added as `R` functions:

```
# providing a bad name will show a warning with valid names:
try <- create_equations("help")
#> Warning in create_equations("help"):
#> Valid flags are: default, li6400, li6800, gfs3000, ciras4,
#> cuticular_conductance, GoffGratch1946, Buck1981, Buck1996,
#> gfs3000_light_bot, gm_fluorescence, d13C, d13C_e_Busch2020, d13C_dis,
#> match, raw, O2_correction
#> Warning in create_equations("help"): No valid equations found!
```

```
# an equation set for gm, with a custom equation for resistance:
create_equations("gm_fluorescence", FLR.rm = \(x) {1 / FLR.gm})
#> $FLR.Cc
#> {
#>     GP <- GasEx.A + g0(Const.RL, NA_real_, "µmol*m^-2*s^-1")
#>     JF <- g0(FLR.ETR, NA_real_, "µmol*m^-2*s^-1")
#>     g0(Const.GammaStar, NA_real_) * (JF + 8 * GP)/(JF - 4 * GP)
#> }
#>
#> $FLR.gm
#> {
#>     GasEx.A/(GasEx.Ci - FLR.Cc)
#> }
#>
#> $FLR.rm
#> {
#>     1/FLR.gm
#> }
#>
```

[6]Pebesma E, Mailund T, Hiebert J (2016) Measurement Units in R. R J 8:486–494. Link

```
#> $gasanalyzer.UseFlags
#> [1] "gm_fluorescence"
```

The calibration of infra-red gas analyzers is sensitive to the gas-mixture used for the measurements. Thus, when doing measurements are done under low oxygen, this has to be taken into account. Analyzers produced by LI-COR Inc. or Heinz Walz GmbH allow an oxygen concentration to be specified or measured and adjust the reported concentrations based on several calibration factors. These equations are unfortunately not included in the stored data files, but are included when either the `O2_correction` or `raw` flags are passed to `create_equations()` as done above. The latter allows for recalculation from raw absorptance values but requires such data to be stored during the measurement, which is not done by default. The `O2_correction` flag back-calculates such raw values based on the original O_2 concentration specified in the data, and then recalculates gas mole fractions based on a potentially updated O_2 concentration. It must be added that since the data reported by the instrument may be averaged before storing, recalculating the data at different oxygen levels may not always be exact.

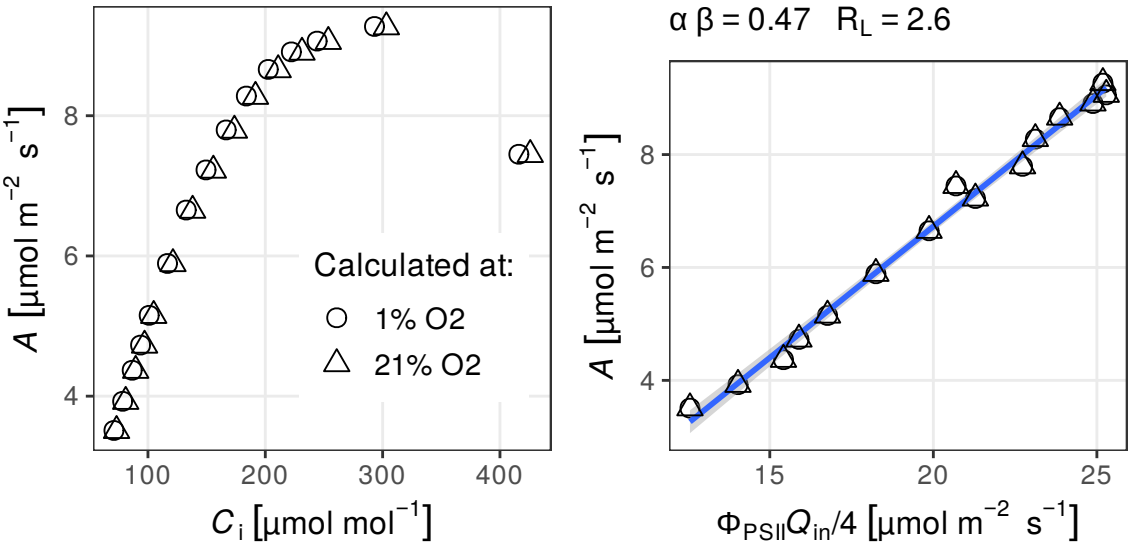
The equations depend on calibration values that depend on the instrument type, and it may be a good idea to verify the calibration factors. For the current example, the calibration factors for the used instrument need to be imported before reading the data (as was mentioned at the top of this section).

To investigate how much the results would differ when a user forgot to enter the correct O_2 concentration during the measurement, I have recalculated the data using an oxygen concentration of 21 %:

```
xlsxfile21 <- xlsxfileRecal |>
  transform(Const.Oxygen = set_units(21, "%")) |>
  recalculate(RecalEqs)
```

And plot the results:

```
AvsCi21 <- xlsxfile21 |> ggplot() +
  geom_point(aes(GasEx.Ci, GasEx.A, shape = "21% O2")) +
  geom_point(aes(xlsxfileRecal$GasEx.Ci,
                 xlsxfileRecal$GasEx.A, shape = "1% O2")) +
  scale_shape_manual(name = "Calculated at:",
                     values = c("1% O2" = 1, "21% O2" = 2)) +
  theme(legend.position = "inside", legend.position.inside = c(0.7, 0.3))

AvsPhiQin21 <-
  xlsxfile21 |>
  lm(GasEx.A ~ FLR.PhiQin_4, data = _) |> (\(x) {
    ggplot(x$model, aes(x = .data[[names(x$model)[2]]],
                        y = .data[[names(x$model)[1]]])) +
      geom_point() +
      stat_smooth(method = "lm", formula = y ~ x) +
      labs(subtitle = bquote(alpha ~ beta == .(x$coef[[2]]) ~~~
                             R[L] == .(-x$coef[[1]]))) +
      geom_point(data = xlsxfileRecal, shape = 2)
    })()

grid.arrange(nice_labels(AvsCi21), nice_labels(AvsPhiQin21), ncol = 2)
```



The figure confirms that since the correction mainly affects water concentrations, variables such as g_{sw} and therefore also C_i are most affected. However the effect on C_i remained below 5%. It is noteworthy that the effect of the oxygen correction is negligible for the assimilation rate, and therefore also does not affect the relation between A and J_F:

```
xlsxfileRecal |>
    lm(GasEx.A ~ I(FLR.ETR / 4), data = _) |> coef()
#>  (Intercept) I(FLR.ETR/4)
#>         -2.6          1.0
```

```
xlsxfile21 |>
    lm(GasEx.A ~ I(FLR.ETR / 4), data = _) |> coef()
#>  (Intercept) I(FLR.ETR/4)
#>         -2.6          1.0
```

2. GFS-3000, loading multiple files and doing a sensitivity analysis

Using base R functions, multiple files can easily be concatenated and used with `gasanalyzer`:

```
aciFiles <- list.files(exampleFiles, "aci\\d\\.csv", full.names = TRUE)

ACi <- lapply(aciFiles, \(x) {print(basename(x)); read_gfs(x, tz = "CEST")}) |>
   do.call("rbind", args = _)
#> [1] "aci1.csv"
#> [1] "aci2.csv"
#> [1] "aci3.csv"
```

```
data.frame(aggregate(list(RH = ACi$GasEx.RHcham),
                     list(file = ACi$SysObs.Filename), mean),
           RH.SD = aggregate(list(ACi$GasEx.RHcham),
                     list(ACi$SysObs.Filename), sd)[, 2]) |>
  knitr::kable()
```

file	RH	RH.SD
aci1	75 [%]	0.64
aci2	74 [%]	1.30
aci3	74 [%]	0.66

The C_i estimates provided by the GFS-3000 instrument do not take boundary layer conductance into account. `gasanalyzer` provides the possibility to recalculate the data using predefined sets of equations. The boundary layer conductance (g_{bw}) of the default chamber at different fan speeds was provided by the instrument vendor (Heinz Walz GmbH). However, the effect on C_i is small when the stomatal conductance (g_s) is much lower than g_{bw}:

```
gfsEqs <- create_equations(c("default", "gfs3000"))
ACi2 <- recalculate(ACi, gfsEqs)
all.equal(ACi$GasEx.Ci, ACi2$GasEx.Ci)
#> [1] "Mean relative difference: 0.0042"
```

The relationship between A_n and C_i can be used to obtain *in vivo* estimates for parameters related to biochemical processes underlying photosynthesis. The R package `photosynthesis`[7] implements methods to obtain such estimates. Recently it has been suggested[8] that the C_i values calculated using equations implemented in most gas-exchange instruments are biased because the underlying theory did not account for gas fluxes through the cuticula. Since estimates for cuticular conductance are not widely available, it can be useful to perform a sensitivity analysis on gas-exchange data to investigate the possible consequences of limiting cuticular conductance. A first assessment of the importance of including cuticular conductance was recently published[9]. Here, an example is shown that includes a similar analysis, but also test for the effect of potential bias in assumed values for the ratio of stomatal conductances between both sides of the leaf (`Const.K`), cuticular conductance (`Const.gcw`), assumed relative humidity in the stomatal cavities (`Const.RHi`), and the measured values for the leaf temperature (`Meas.Tleaf`) and H_2O mol fraction reported by the instrument (`Meas.H2Or`). The `permutate()` method was used to create a range of values for these quantities while keeping all other measured and assumed variables constant. Subsequently all derived variables are recomputed:

```
# we define some constants that specify the relative value of the variables
# on which we will do a sensitivity analysis:
SAset <- c("Const.calk", "Const.calg", "Const.calrhi", "Const.calt",
           "Const.calh")
# symbols for the plot header:
SAlabs <- c("italic(K)", "italic(g)[cw]", "RH[i]",
            "italic(T)[leaf]", "'['*H[2]*O*']'[r]")
# set defaults to 1 (100%) and define sequences over which to permutate
ACi[SAset] <- 1
# a 5% range:
seq5 <- seq(-0.05, 0.05, length.out = 11)
seq5 <- seq5[seq5 != 0]

# make equations allowing to vary the variables by a relative value:
MSFeqs <- create_equations(c("default", "gfs3000", "cuticular_conductance"),
                           Const.K = function() { Const.calk * Const.K },
                           Meas.Tleaf = function() { Const.calt * Meas.Tleaf },
                           Const.gcw = function() { Const.calg * Const.gcw },
                           Const.gcc = function() { Const.gcw / 20 },
                           Const.RHi = function() { Const.calrhi * Const.RHi },
                           Meas.H2Or = function() { Const.calh * Meas.H2Or },
                           Meas.H2Os = function() { Const.calh * Meas.H2Os })

# set gcw and use steady-state equations, then use the sequences
```

[7] Stinziano JR, Roback C, Sargent D, et al (2021) Principles of resilient coding for plant ecophysiologists. AoB PLANTS 13:plab059. Link

[8] Márquez DA, Stuart-Williams H, Farquhar GD (2021) An improved theory for calculating leaf gas exchange more precisely accounting for small fluxes. Nat Plants 7:317–326. Link

[9] Lamour J, Davidson KJ, Ely KS, et al (2022) New calculations for photosynthesis measurement systems: what's the impact for physiologists and modelers? New Phytol 233:592–598. Link

```
ACi[["Const.gcw"]] <- set_units(25, "mmol*m^-2*s^-1")
ACi[["Const.RHi"]] <- set_units(95, "%")
ACi[["SysConst.UseDynamic"]] <- FALSE

ACi.SA <- rbind(permutate(ACi, Const.calk = c(1, 1 + 20 * seq5)),
                permutate(ACi, Const.calg = 1 + 20 * seq5),
                permutate(ACi, Const.calrhi = 1 + seq5),
                permutate(ACi, Const.calt = 1 + seq5),
                permutate(ACi, Const.calh = 1 + seq5)) |>
  recalculate(MSFeqs)
```

In the next step, the `photosynthesis` package was used to obtain estimates for the carboxylation capacity (V_{cmax}), electron transport rate (J), triose-phosphate limitation (T_p) and the respiration rate in the light (R_L). To facilitate analysis of the results, the following function takes care of some intermediate steps with the curve-fitting and provides convenient tabular output:

```
library(photosynthesis)
#> Loading required package: minpack.lm
```

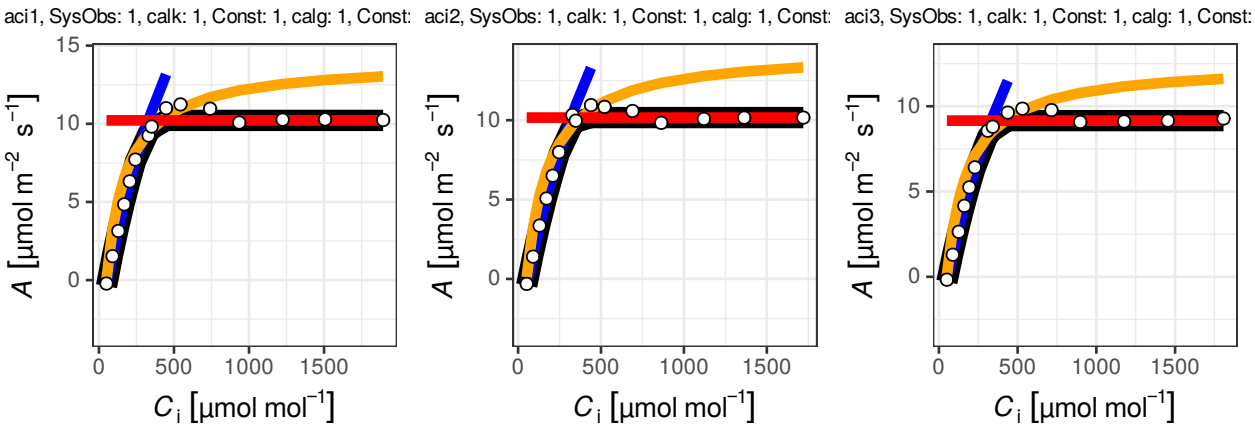
```
aPars <- c("V_cmax", "J_max", "V_TPU", "R_d")

fitaci_per_group <- function(df, aPars, g1, ...) {
  grp <- c(g1, ...)
  outCols <- c(grp, aPars, "group", "Plots")
  splitOn <- paste("~", paste(grp, collapse = " + "))
  # drop category name and Filename for shorter labels:
  lbls <- vapply(strsplit(grp, ".", fixed = TRUE), \(x) rev(x), c("nm","gp")) |>
    paste0(": ") |>
    gsub("Filename: ", "", x = _, fixed = TRUE)
  # layout:
  plotAdd <- list(do.call(labs, var2label(list(y = "GasEx.A", x = "GasEx.Ci"),
                                          TRUE)),
                  theme(legend.position = "none", plot.title.position = "plot",
                        plot.title = element_text(size = 8)))
  # translate names to what the photosynthesis package uses (sadly not essdive):
  out <- df[c("GasEx.A", "GasEx.Ci", "LeafQ.Qin", "Meas.Tleaf",
              "Meas.Pa", "gasanalyzer.Permutate", grp)] |>
    setNames(c("A_net", "C_i", "PPFD", "T_leaf", "Pa", "group", grp)) |>
    drop_units() |> transform(T_leaf = T_leaf + 273.15) |>
    split(as.formula(splitOn), drop = TRUE) |>
    lapply(\(x) {
      lbl <- paste0(lbls, x[1, grp], collapse=", ")
      fit <- try(fit_aci_response(x, Pa = mean(x$Patm)))
      if (!inherits(fit, "try-error"))
        data.frame(x[1, c("group", grp)],
                   fit[["Fitted Parameters"]],
                   Plots = I(list(fit$Plot + ggtitle(lbl) + plotAdd))
                   )[outCols]
      else
        data.frame()
    })
  do.call(rbind, c(out, make.row.names = FALSE))
}
```

Now, we will call this function and transform the output to relative values and average them over the 3 plants:

```
# call the function to fit the curves:
CO2curves.SA <- ACi.SA |>
  fitaci_per_group(aPars, "SysObs.Filename", SAset)

# reference values are where the SAset variables equal 1:
refids <- rowSums(CO2curves.SA[SAset] == 1) == length(SAset)
# combine all relative values in one column based on group:
CO2curves.SA$gval <- unlist(CO2curves.SA[cbind(seq_len(nrow(CO2curves.SA)),
                                       match(CO2curves.SA$group,
                                             names(CO2curves.SA)))])
# value as relative change:
CO2curves.SA$gval <- 100 * (as.numeric(CO2curves.SA$gval) - 1)

# calc mean over the 3 reps:
CO2curves.SAm <-
  lapply(names(CO2curves.SA[aPars]),
         \(x) {
           # relative change:
           rval <- 100 * as.numeric((CO2curves.SA[[x]] -
                                       CO2curves.SA[[x]][refids]) /
                                      abs(CO2curves.SA[[x]][refids]))
           # grouping:
           byl <- as.list(CO2curves.SA[SAset])
           with(CO2curves.SA,
                data.frame(nm = x,
                           aggregate(cbind(gval), byl, head, n = 1),
                           group = aggregate(group, byl, head, n = 1)[["x"]],
                           val = aggregate(rval, byl, mean)[["x"]],
                           sd = aggregate(rval, byl, sd)[["x"]]))}) |>
  do.call(rbind, args = _)
```

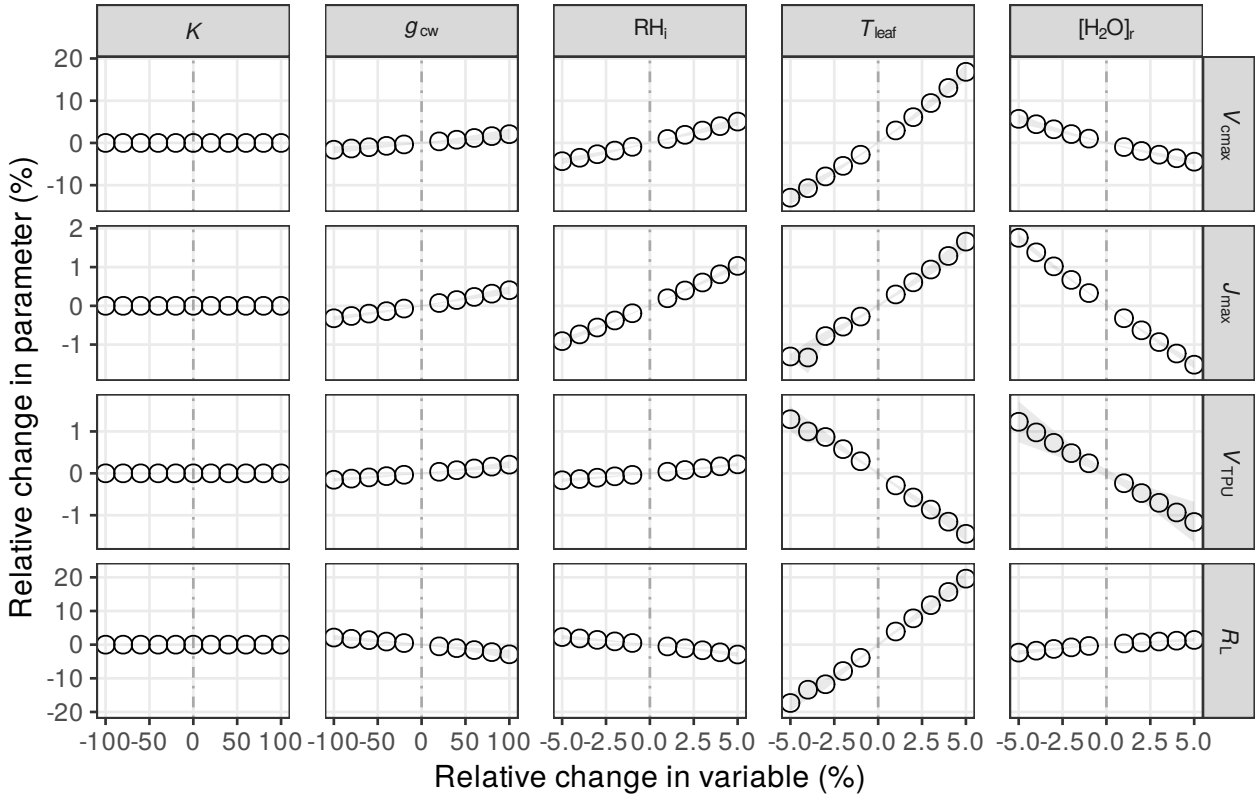
The model fits computed by the `photosynthesis` package are stored in the data and can be plotted:

```
grid.arrange(grobs = CO2curves.SA$Plots[refids], ncol = 3)
```



The code below creates a combined plot showing how a relative change in the 5 different variables affects the parameters of the C_3 biochemical model:

```
# make lbls factors for good order and plotmath
CO2curves.SAm$group <- factor(CO2curves.SAm$group,
                              levels = SAset,
                              labels = SAlabs)
CO2curves.SAm$nm <- factor(CO2curves.SAm$nm,
                           levels = c("V_cmax", "J_max", "V_TPU", "R_d"),
                           labels = c("italic(V)[cmax]", "italic(J)[max]",
                                      "italic(V)[TPU]", "italic(R)[L]"))
# and plot:
CO2curves.SAm |>
  ggplot(aes(x = gval, y = val, group = nm)) +
  geom_vline(aes(xintercept = 0), linetype = "dotdash", color = "darkgrey") +
  geom_point() +
  geom_ribbon(aes(ymin = val - sd, ymax = val + sd), alpha = 0.1) +
  facet_grid(cols = vars(group), rows = vars(nm),
             scales="free", labeller = label_parsed) +
  xlab("Relative change in variable (%)") +
  ylab("Relative change in parameter (%)") +
  theme(plot.title = element_text(size = 14),
        panel.grid.minor = element_blank(),
        panel.spacing.x = unit(14, "pt"))
```



The results show that for these measurements, accounting for stomatal or cuticular conductance has only very small effects on the estimates. Bias in leaf temperature measurements have much larger effects.

3. Combining gas-exchange data with isotope discrimination

The ability to quick analyze the sensitivity of a result to different constants is also valuable for calculations of mesophyll conductance (g_m) from concurrent measurements of gas exchange and carbon isotope discrimination. As an example, a data file from an experiment combining gas exchange with carbon isotope measurements was included. This file contains additional columns with the $\delta^{13}C$ value of the reference and sample air, and of the air in the glasshouse where the plants were grown.

The dataset can be used to estimate the mesophyll conductance. Such calculations require several fractionation factors describing how the abundance ratio of the carbon isotopes will change using a physiological or physical process. Since uncertainty exist surrounding the exact value of such factors, it is useful to analyze the data under several different assumptions. In addition, the effect of different assumptions in the isotopic model (see Busch et al. 2020[10]) can be tested.

In the following example, `gasanalyzer` is used to analyze the effect of the fractionation factor associated with photorespiration (f) using both the connected and disconnected isotopic models as described by Busch et al. 2020.

```
# load data
isoFile <- file.path(exampleFiles, "d13C.tsv")
# read and recalculate
isotopes <- read_gasexchange(isoFile) |>
  recalculate(create_equations(c("default", "li6400")))

# vary f between 0 and 20, and use 2 sets of equations:
isotopes$d13CConst.e <- set_units(0, "permille")
isotopes$gasanalyzer.Equations <- list(NA)
isotopes <- isotopes |>
  permutate(d13CConst.f = seq(0, 20, 0.5)) |>
  permutate(gasanalyzer.Equations =
              c(connected = list(create_equations("d13C")),
                disconnected = list(create_equations(c("d13C",
                                                       "d13C_dis"))))) |>
  recalculate() |>
  transform(factor.f = factor(d13CConst.f))

# point method gm is already calculated, average the results:
gmEst <- aggregate(list(gm = as.numeric(isotopes$d13C.gm)),
                   list(d13CConst.f = isotopes$d13CConst.f,
                        Model = isotopes$gasanalyzer.PermutateLabel),
                   FUN = function(x)
                     c(d13C.gm = mean(x), d13C.gm.sem = sd(x)
                       / sqrt(length(x))))
gmEst <- do.call(data.frame, c(gmEst[1:2], unname(gmEst[3])))
gmEst$method <- "point"
```

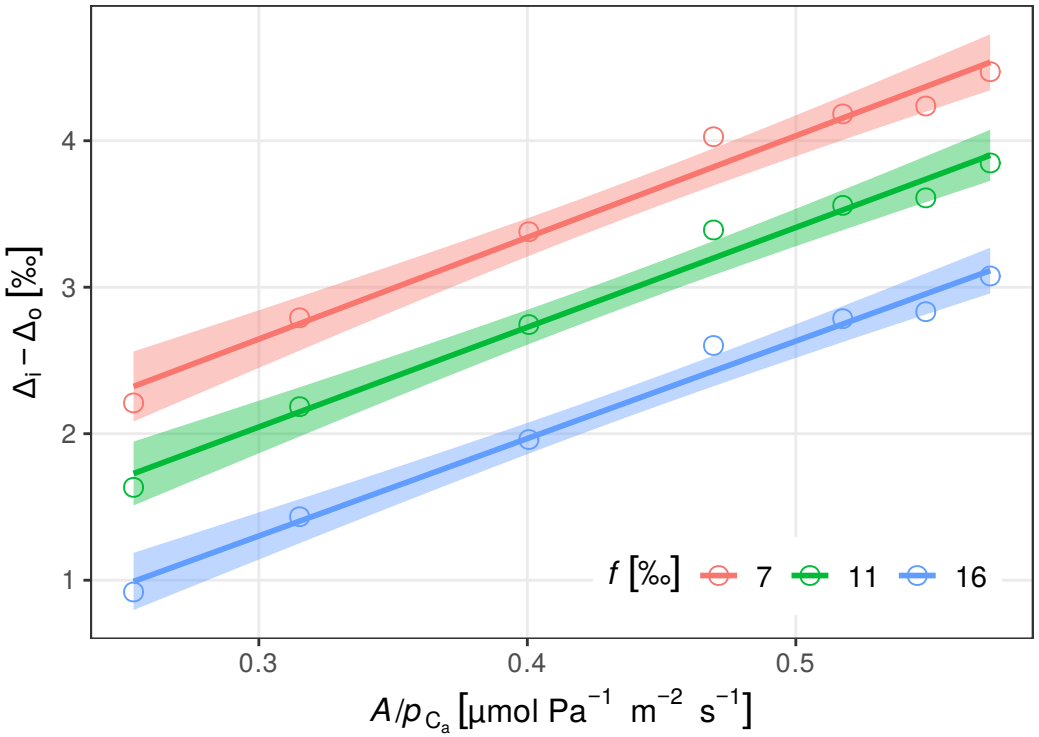
Before plotting the results, it is useful to also derive g_m estimates using the so-called slope method (see e.g. Tazoe et al. 2009[11]. For this approach, the relationship between the difference between modeled and observed discrimination is plotted against the ratio of net assimilation and the partial pressure of CO_2 in the ambient air (A/p_{C_a}):

```
d13Cslopes <- isotopes |>
```

[10] Busch FA, Holloway-Phillips M, Stuart-Williams H, Farquhar GD (2020) Revisiting carbon isotope discrimination in C_3 plants shows respiration rules when photosynthesis is low. Nat Plants 6:245–258. Link

[11] Tazoe Y, von Caemmerer S, Badger MR, Evans JR (2009) Light and CO2 do not affect the mesophyll conductance to CO2 diffusion in wheat leaves. J Exp Bot 60:2291–2301. Link

```
  subset(gasanalyzer.PermutateLabel == "connected" &
           as.numeric(d13CConst.f) %in% c(7, 11, 16)) |>
  ggplot(aes(y = d13C.DeltaiDeltao, x = d13C.A_pCa, color = factor.f)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, aes(fill = factor.f)) +
    guides(fill = "none",
         colour = guide_legend(override.aes = list(fill = NA))) +
  labs(color = var2label("d13CConst.f", TRUE)[[1]]) +
  theme(legend.position = "inside", legend.position.inside = c(0.75, 0.10),
        legend.direction = "horizontal")

nice_labels(d13Cslopes)
```



The mesophyll conductance is inversely related to the slope of this relationship. The slopes appear quite similar, suggesting that the slope method is much less sensitive to assumptions on the value taken for f. However, a more thorough analysis requires a nonlinear fitting approach.

Such an approach has an additional advantage, because it allows for the estimation of the effective fractionation associated with mitochondrial respiration (e'). Respiration typically releases carbon from substrate that was assimilated before the leaf was clamped in the gas-exchange chamber, and since the isotopic composition of the tank air typically differs drastically from that of the air during growth conditions, this may result in a strong isotopic signal that needs to be accounted for. The correction proposed by Busch et al. 2020 could not be used for this analysis because the discrimination against ^{13}C under growth conditions was not quantified for these plants. However, for the slope method, it is possible to fit both e' and g_m in one analysis (Tazoe et al. 2009), making the results free of any assumptions regarding the actual effect of respiratory fractionations.

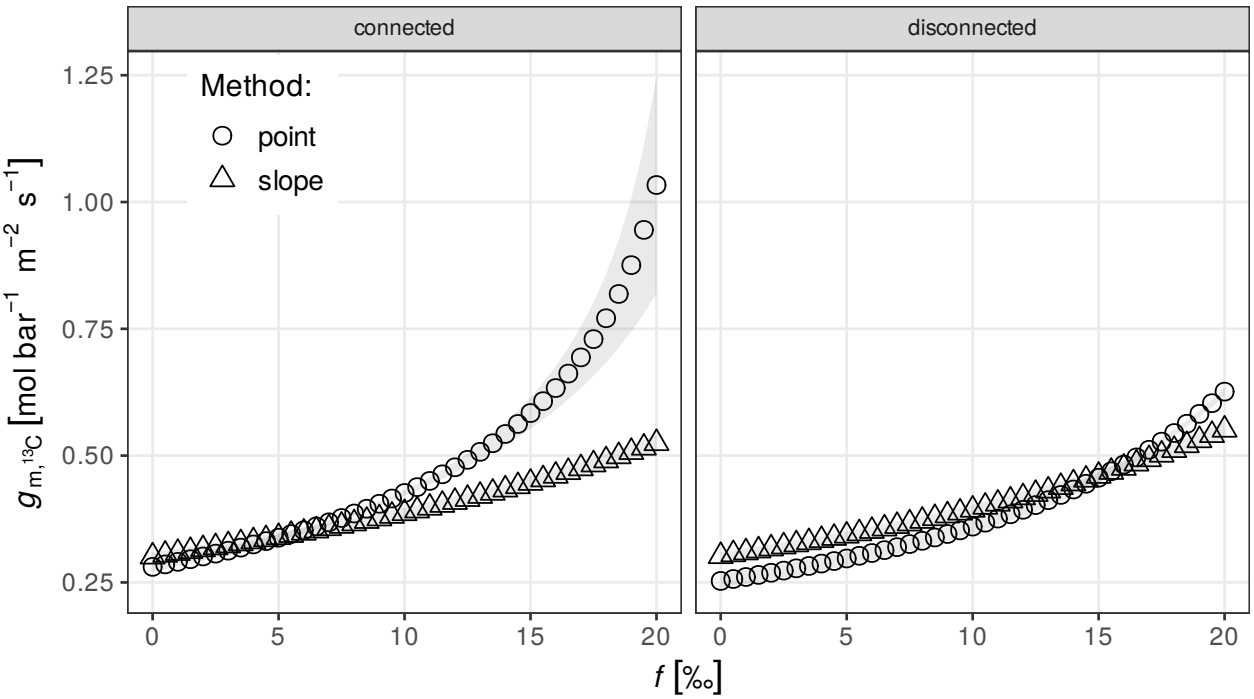
Below we will use the isotope equations for the connected and disconnected model provided by `gasanalyzer` for the estimation of g_m using the slope method:

```
# The equation for the slope method is already in the data, so this function
# just evaluates that. Its not the fastest way, because recalculate adds
# overhead and unneccessary calcs, but simple:
```

```
fitfunc <- function(gm, ep) {
    # Di in the equation doesn't take gm into account, and assumes Ci=Cc
    # to get the true Delta we so we modify Ci to be Cc again:
    df$GasEx.Ci <-  df$GasEx.Ci - df$GasEx.A /
      (set_units(gm, "mol*m^-2*s^-1*bar^-1") * df$Meas.Pa)
    # substitute estimated ep in data and recalc:
    df$d13C.ep <- set_units(ep, "permille")
    as.numeric(recalculate(df)$d13C.Deltai)
}

#this will take a while:
gmEstSlope <- isotopes |>
  #we want to fit ep, so modify the equation list to not recalculate it:
  transform(gasanalyzer.Equations =
              lapply(gasanalyzer.Equations,
                     \(x) modifyList(x, list(d13C.ep = NULL)))) |>
  #split by f and equation version, and call nls:
  split(~ isotopes$d13CConst.f + isotopes$gasanalyzer.PermutateLabel) |>
  lapply(function(dat) {
           # data is moved to environment of fitfunc
           environment(fitfunc) <- list2env(list(df = dat))
           nlsfit <- nls(drop_units(d13C.Deltao) ~ fitfunc(gm, ep),
                         start = c(gm = 0.3, ep = -10), dat)
           cfs <- summary(nlsfit)$coefficients
           data.frame(d13CConst.f = dat$d13CConst.f[1],
                      Model = dat$gasanalyzer.PermutateLabel[1],
                      d13C.gm = cfs[1],
                      d13C.gm.sem = cfs[3], d13C.ep = cfs[2],
                      d13C.ep.sem = cfs[4])}) |>
  do.call(rbind, args = _) |>
  transform(method = "slope")

gmEst <- rbind(gmEst, gmEstSlope[names(gmEstSlope) %in% names(gmEst)],
               make.row.names = FALSE)

#re-adds units:
units(gmEst$d13C.gm) <- deparse_unit(isotopes$d13C.gm)
units(gmEst$d13C.gm.sem) <- deparse_unit(isotopes$d13C.gm)
```

Plot the results of the analysis:

```
fsens <- ggplot(gmEst, aes(d13CConst.f, d13C.gm,
                           ymin = d13C.gm - d13C.gm.sem,
                           ymax = d13C.gm + d13C.gm.sem,
                           shape = method, color = Model)) +
  geom_point(color = "black") +
  geom_ribbon(alpha = 0.1, color = NA) + guides(color = "none") +
  scale_shape_manual("Method:", values = c(1, 2)) + facet_wrap(vars(Model)) +
  theme(legend.position = "inside", legend.position.inside = c(0.12, 0.85),
        panel.grid.minor = element_blank())
nice_labels(fsens)
```



The results indicate that the disconnected model resulted in small differences between the slope and point methods. Although all approaches showed some increase in the g_m estimate when higher values for photorespiratory fractionation f were used, the connected model lead to an unusual strong increase in the estimates when higher values for f were used.