

# Package: fluxfinder (via r-universe)

January 10, 2025

**Type** Package

**Title** Parsing, Computation, and Diagnostics for Greenhouse Gas Measurements

**Version** 1.0.0

**Description** Parse static-chamber greenhouse gas measurement files generated by a variety of instruments; compute flux rates using multi-observation metadata; and generate diagnostic metrics and plots. Designed to be easy to integrate into reproducible scientific workflows.

**URL** <https://github.com/COMPASS-DOE/fluxfinder>

**License** MIT + file LICENSE

**Encoding** UTF-8

**Imports** broom (>= 1.0), jsonlite (>= 1.8), lubridate (>= 1.0), MASS (>= 7.0)

**Suggests** gasfluxes, ggplot2, knitr, rmarkdown, testthat (>= 3.0.0), withr (>= 2.0)

**Config/testthat/edition** 3

**RoxygenNote** 7.3.1

**VignetteBuilder** knitr

**BugReports** <https://github.com/COMPASS-DOE/fluxfinder/issues>

**NeedsCompilation** no

**Author** Stephanie Wilson [cre] (<https://orcid.org/0000-0002-5484-0748>), Ben Bond-Lamberty [aut] (<https://orcid.org/0000-0001-9525-4633>), Genevieve Noyce [ctb] (<https://orcid.org/0000-0003-0423-6478>), Roberta Bittencourt Peixoto [ctb] (<https://orcid.org/0000-0002-8053-2730>), Patrick Megonigal [ctb] (<https://orcid.org/0000-0002-2018-7883>), Smithsonian Institution [cph, fnd]

**Maintainer** Stephanie Wilson <[sjw22120@gmail.com](mailto:sjw22120@gmail.com)>

**Date/Publication** 2024-10-22 13:20:02 UTC

**Additional\_repositories** <https://cranhaven.r-universe.dev>

**Config/pak/sysreqs** libicu-dev

**Repository** <https://cranhaven.r-universe.dev>

**RemoteUrl** <https://github.com/cranhaven/cranhaven.r-universe.dev>

**RemoteRef** package/fluxfinder

**RemoteSha** 3d93253752f1b45df0019cf634a1c276cd701294

**RemoteSubdir** fluxfinder

## Contents

ffi_compute_fluxes . . . . .	2
ffi_fit_models . . . . .	3
ffi_hm1981 . . . . .	5
ffi_metadata_match . . . . .	6
ffi_qaqc . . . . .	7
ffi_read_EGM4 . . . . .	8
ffi_read_LGR915 . . . . .	8
ffi_read_LI7810 . . . . .	9
ffi_read_LI7820 . . . . .	10
ffi_read_LI850 . . . . .	10
ffi_read_LIsmartchamber . . . . .	11
ffi_read_PicarroG2301 . . . . .	12
ideal-gas-law . . . . .	12

**Index** **14**

---

ffi\_compute\_fluxes      *Compute fluxes for multiple groups (measurements)*

---

## Description

Compute fluxes for multiple groups (measurements)

## Usage

```
ffi_compute_fluxes(
  data,
  group_column,
  time_column,
  gas_column,
  dead_band = 0,
  normalize_time = TRUE,
  fit_function = ffi_fit_models,
  ...
)
```

## Arguments

data	A <code>data.frame</code> (or tibble or data.table)
group_column	Name of the grouping column in data, character; pass <code>NULL</code> to run with no grouping
time_column	Name of the time column in data, character
gas_column	Name of the gas (concentration or quantity) column in data, character
dead_band	Length of dead band, the equilibration period at the beginning of the time series during which data are ignore, in seconds (numeric)
normalize_time	Normalize the values so that first is zero? Logical
fit_function	Optional flux-fit function; default is <code>ffi_fit_models</code>
...	Other parameters passed to <code>fit_function</code>

## Value

A `data.frame` with one row per `group_column` value. It will always include the mean, minimum, and maximum values of `time_column` for that group, but other columns depend on what is returned by the `fit_function`.

## See Also

[ffi\\_fit\\_models](#)

## Examples

```
# No grouping
ffi_compute_fluxes(cars, group_column = NULL, "speed", "dist")
# With grouping
cars$Plot <- c("A", "B")
ffi_compute_fluxes(cars, "Plot", "speed", "dist")
# See the introductory vignette for a fully-worked example with real data
```

---

ffi\_fit\_models

*Fit various models to gas concentration data*

---

## Description

Fit various models to gas concentration data

## Usage

```
ffi_fit_models(time, conc, area, volume)
```

**Arguments**

time	Relative time of observation (typically seconds), numeric
conc	Greenhouse gas concentration (typically ppm or ppb), numeric
area	Area covered by the measurement chamber (typically cm <sup>2</sup> ), numeric
volume	Volume of the system (chamber + tubing + analyzer, typically cm <sup>3</sup> ), numeric

**Details**

If a linear model cannot be fit, NULL is returned. If the robust linear and/or polynomial models cannot be fit, then NA is returned for their particular statistics. The HM1981 approach is only valid for saturating (exponential) data and NA is returned otherwise.

**Value**

A wide-form `data.frame` with fit statistics for linear ("lin", `lm`), robust linear ("rob", `r1m`), polynomial ("poly"), and H&M1981 ("HM81", `ffi_hm1981`) models. The latter is based on an exponential model drawn from one-dimensional diffusion theory; see Hutchinson and Mosier (1981) and Nakano et al. (2004).

For each model type, the following columns are returned:

- Model statistics AIC, r.squared, RMSE, and p.value;
- Flux (slope) statistics `flux.estimate` and `flux.std.error`;
- Intercept statistics `int.estimate` and `int.std.error`;
- For the robust linear regression model only, a logical value `converged`.

**Note**

Normally this is not called directly by users, but instead via `ffi_compute_fluxes`.

**References**

Nakano, T., Sawamoto, T., Morishita, T., Inoue, G., and Hatano, R.: A comparison of regression methods for estimating soil-atmosphere diffusion gas fluxes by a closed-chamber technique, *Soil Biol. Biochem.*, 36, 107–113, 2004. doi:10.1016/j.soilbio.2003.07.005

Hutchinson, G. L. and Mosier, A. R.: Improved soil cover method for field measurement of nitrous oxide fluxes, *Soil Sci. Soc. Am. J.*, 45, 311-316, 1981. doi:10.2136/sssaj1981.03615995004500020017x

**Examples**

```
# Toy data - linear
ffi_fit_models(cars$speed, cars$dist)

# Toy data - nonlinear
ffi_fit_models(Puromycin$conc, Puromycin$rate)

# Real data
f <- system.file("extdata/TG10-01087.data", package = "fluxfinder")
dat <- ffi_read_LI7810(f)[1:75,] # isolate first observation
```

```
dat$SECONDS <- dat$SECONDS - min(dat$SECONDS) # normalize time to start at 0
plot(dat$SECONDS, dat$CO2)
ffi_fit_models(dat$SECONDS, dat$CO2)
```

---

ffi\_hm1981

*Compute flux using nonlinear Hutchinson and Mosier (1981) model*

---

## Description

Compute flux using nonlinear Hutchinson and Mosier (1981) model

## Usage

```
ffi_hm1981(time, conc, h = 1)
```

## Arguments

time	Time values, numeric
conc	Gas concentration values, numeric
h	Effective chamber height

## Value

Flux estimate; see references for more information.

## References

Hutchinson, G. L. and Mosier, A. R.: Improved soil cover method for field measurement of nitrous oxide fluxes, Soil Sci. Soc. Am. J., 45, 311-316, 1981. doi:[10.2136/sssaj1981.03615995004500020017x](https://doi.org/10.2136/sssaj1981.03615995004500020017x)

## Examples

```
# If data are approximately linear, then NA is returned
ffi_hm1981(cars$speed, cars$dist)
# If data are nonlinear (saturating) then flux based on gas diffusion theory
ffi_hm1981(Puromycin$conc, Puromycin$rate)
```

---

```
ffi_metadata_match      Match metadata info with a vector of data timestamps
```

---

## Description

Match metadata info with a vector of data timestamps

## Usage

```
ffi_metadata_match(data_timestamps, start_dates, start_times, obs_lengths)
```

## Arguments

<code>data_timestamps</code>	Data timestamps, either character (YYYY-MM-DD HH:MM:SS) or <a href="#">POSIXct</a>
<code>start_dates</code>	Metadata measurement dates, either character (YYYY-MM-DD) or <a href="#">POSIXct</a>
<code>start_times</code>	Metadata measurement start time entries, either character (HH:MM:SS) or <a href="#">period</a>
<code>obs_lengths</code>	Observation lengths in seconds, numeric; must be same length as <code>start_dates</code> . This should include both the intended measurement period as well as any dead band time at the beginning

## Value

A numeric vector equal in length to `data_timestamps`, with each entry indicating the metadata entry that should be used for that observation. NA is returned if a timestamp has no match in the metadata (i.e., does not fall within any window defined by the `start_dates`, `start_times`, and observation length parameters).

## Note

If `data_timestamps` or `start_dates` cannot be parsed as YYYY-MM-DD, the preferred format, then MM/DD/YYYY (used by U.S. versions of Microsoft Excel when saving CSV files, for example) will be tried.

## Examples

```
# Data timestamps
d_t <- c("2024-01-01 13:00:05", "2024-01-01 13:00:10",
"2024-01-01 13:05:05", "2024-01-01 13:10:00")
# Metadata start dates and times: two measurements, starting 5 minutes apart
s_d <- c("2024-01-01", "2024-01-01")
s_t <- c("13:00:00", "13:05:00")
ol <- c(60, 60) # Observation lengths
ffi_metadata_match(d_t, s_d, s_t, ol)
# Returns {1, 1, 2, NA} indicating that the first and second data timestamps
# correspond to metadata entry 1, the third to entry 2, and the fourth
# has no match
```

```
# This generates an error because of overlapping timestamps:
## Not run:
s_t <- c("13:00:00", "13:01:00")
ffi_metadata_match(d_t, s_d, s_t, ol)

## End(Not run)
```

---

ffi\_qaqc

*Generate a QA/QC document*

---

## Description

Generate a QA/QC document

## Usage

```
ffi_qaqc(
  flux_data,
  group_column,
  output_file = "qaqc.html",
  output_dir = getwd(),
  open_output = TRUE
)
```

## Arguments

flux_data	A data frame from <a href="#">ffi_compute_fluxes</a> or similar
group_column	Name of the grouping label column in flux_data, character; pass NULL to run with no grouping
output_file	Name of the output file
output_dir	Name of the output directory; default is current working directory
open_output	Automatically open the output HTML file?

## Value

The path of the output file.

## Examples

```
# Toy data
cars$Plot <- c("A", "B")
fd <- ffi_compute_fluxes(cars, "Plot", "speed", "dist")
x <- ffi_qaqc(fd, group_column = "Plot")
file.remove(x) # clean up
# See the introductory vignette for a fully-worked example with real data
```

---

ffi_read_EGM4	<i>Read an EGM-4 data file</i>
---------------	--------------------------------

---

**Description**

Read an EGM-4 data file

**Usage**

```
ffi_read_EGM4(file, year, tz = "UTC")
```

**Arguments**

file	Filename to read, character
year	Four-digit year of the data (EGM-4 output files have month, day, hour, and minute, but not year), numeric or character
tz	Time zone of the file's time data, character (optional)

**Value**

A `data.frame` with the parsed data.

**Examples**

```
f <- system.file("extdata/EGM4-data.dat", package = "fluxfinder")
dat <- ffi_read_EGM4(f, 2023)
dat <- ffi_read_EGM4(f, 2023, tz = "EST") # specify time zone
```

---

ffi_read_LGR915	<i>Read a LGR 915-0011 data file</i>
-----------------	--------------------------------------

---

**Description**

Read a LGR 915-0011 data file

**Usage**

```
ffi_read_LGR915(file, date_format = "DMY", tz = "UTC")
```

**Arguments**

file	Filename to read, character
date_format	Date format, character: "MDY" (month-day-year) "DMY" (day-month-year), or "YMD" (year-month-day)
tz	Time zone of the file's time data, character (optional)



**Details**

The LGR 915-0011 was an Ultra-Portable Greenhouse Gas Analyzer made by Los Gatos Research. The date in its output files can appear in different formats, which is why the `date_format` parameter is needed.

**Value**

A `data.frame` with the parsed data.

**Note**

Some LGR 915 files can have a PGP block at the end; this is ignored.

**Examples**

```
f <- system.file("extdata/LGR-data.csv", package = "fluxfinder")
dat <- ffi_read_LGR915(f, date_format = "MDY")
dat <- ffi_read_LGR915(f, date_format = "MDY", tz = "EST") # specify time zone
```

---

<code>ffi_read_LI7810</code>	<i>Read a LI-7810 data file</i>
------------------------------	---------------------------------

---

**Description**

Read a LI-7810 data file

**Usage**

```
ffi_read_LI7810(file)
```

**Arguments**

<code>file</code>	Filename to read, character
-------------------	-----------------------------

**Details**

Currently LI-7810 and LI-7820 files are handled identically.

**Value**

A `data.frame` with the parsed data.

**Examples**

```
f <- system.file("extdata/TG10-01087.data", package = "fluxfinder")
dat <- ffi_read_LI7810(f)
```

---

ffi\_read\_LI7820      *Read a LI-7820 data file*

---

**Description**

Read a LI-7820 data file

**Usage**

```
ffi_read_LI7820(file)
```

**Arguments**

file                  Filename to read, character

**Details**

Currently LI-7810 and LI-7820 files are handled identically.

**Value**

A `data.frame` with the parsed data.

**Examples**

```
f <- system.file("extdata/TG20-01182.data", package = "fluxfinder")
dat <- ffi_read_LI7820(f)
```

---

ffi\_read\_LI850      *Read a LI-850 data file*

---

**Description**

Read a LI-850 data file

**Usage**

```
ffi_read_LI850(file, tz = "UTC")
```

**Arguments**

file                  Filename to read, character  
tz                    Time zone of the file's time data, character (optional)

**Value**

A `data.frame` with the parsed data, including a `TIMESTAMP` column.

**Examples**

```
f <- system.file("extdata/LI850.txt", package = "fluxfinder")
dat <- ffi_read_LI850(f)
dat <- ffi_read_LI850(f, tz = "EST") # specify time zone
```

---

ffi\_read\_LIsmartchamber

*Read a LI-8200-01S (smart chamber) data file*

---

**Description**

Read a LI-8200-01S (smart chamber) data file

**Usage**

```
ffi_read_LIsmartchamber(file, concentrations = TRUE)
```

**Arguments**

file                   Filename to read, character  
concentrations       Return concentration data (the default), or just summary information? Logical

**Value**

A `data.frame` with the parsed data.

**Note**

These files are in **JSON** format. See also <https://www.licor.com/env/products/soil-flux/smart-chamber>.

**Author(s)**

Ben Bond-Lamberty

**Examples**

```
f <- system.file("extdata/LI8200-01S.json", package = "fluxfinder")
dat <- ffi_read_LIsmartchamber(f) # returns 240 rows
ffi_read_LIsmartchamber(f, concentrations = FALSE) # only 4 rows
```

---

ffi\_read\_PicarroG2301 *Read a Picarro G2301 data file*

---

**Description**

Read a Picarro G2301 data file

**Usage**

```
ffi_read_PicarroG2301(file, tz = "UTC")
```

**Arguments**

file	Filename to read, character
tz	Time zone of the file's time data, character (optional)

**Value**

A `data.frame` with the parsed data.

**References**

[https://www.picarro.com/environmental/products/g2301\\_gas\\_concentration\\_analyzer](https://www.picarro.com/environmental/products/g2301_gas_concentration_analyzer)

**Examples**

```
f <- system.file("extdata/PicarroG2301-data.dat", package = "fluxfinder")
dat <- ffi_read_PicarroG2301(f)
dat <- ffi_read_PicarroG2301(f, tz = "EST") # specify time zone
```

---

ideal-gas-law

*Convert gas concentration to quantity using the Ideal Gas Law*

---

**Description**

Convert gas concentration to quantity using the Ideal Gas Law

**Usage**

```
ffi_ppm_to_umol(ppm, volume, temp, atm)
```

```
ffi_ppb_to_nmol(ppb, volume, temp, atm)
```

**Arguments**

ppm	Gas concentration (ppmv), numeric
volume	System volume (chamber + tubing + analyzer, m3), numeric
temp	Optional chamber temperature (degrees C), numeric
atm	Optional atmospheric pressure (Pa), numeric
ppb	Gas concentration (ppbv), numeric

**Value**

The quantity value, in micromoles (for `ffi_ppm_to_umol`) or nanomoles (for `ffi_ppb_to_nmol`).

**Note**

If `temp` and/or `atm` are not provided, the defaults are NIST normal temperature and pressure.

**References**

Steduto et al.: Automated closed-system canopy-chamber for continuous field-crop monitoring of CO<sub>2</sub> and H<sub>2</sub>O fluxes, *Agric. For. Meteorol.*, 111:171-186, 2002. [doi:10.1016/S01681923\(02\)00023-0](https://doi.org/10.1016/S01681923(02)00023-0)

**Examples**

```
ffi_ppm_to_umol(400, 0.1)
ffi_ppb_to_nmol(400, 0.1)
```

# Index

`data.frame`, [3](#), [4](#), [8–12](#)

`ffi_compute_fluxes`, [2](#), [4](#), [7](#)  
`ffi_fit_models`, [3](#), [3](#)  
`ffi_hm1981`, [4](#), [5](#)  
`ffi_metadata_match`, [6](#)  
`ffi_ppb_to_nmol (ideal-gas-law)`, [12](#)  
`ffi_ppm_to_umol (ideal-gas-law)`, [12](#)  
`ffi_qaqc`, [7](#)  
`ffi_read_EGM4`, [8](#)  
`ffi_read_LGR915`, [8](#)  
`ffi_read_LI7810`, [9](#)  
`ffi_read_LI7820`, [10](#)  
`ffi_read_LI850`, [10](#)  
`ffi_read_LIsmartchamber`, [11](#)  
`ffi_read_PicarroG2301`, [12](#)

`ideal-gas-law`, [12](#)

`lm`, [4](#)

`period`, [6](#)  
`POSIXct`, [6](#)

`rlm`, [4](#)