

# Package: measure (via r-universe)

October 13, 2024

**Title** A Recipes-style Interface to Tidymodels for Analytical Measurements  
**Version** 0.0.1.9000  
**Description** Analytical measurements...  
**License** MIT + file LICENSE  
**URL** <https://jameshwade.github.io/measure>,  
<https://github.com/jameshwade/measure>  
**BugReports** <https://github.com/jameshwade/measure/issues>  
**Depends** R (>= 3.5.0), recipes  
**Imports** cli, dials, dplyr, generics, glue, IDPmisc, purrr, rlang, tibble, tidyr, vctrs  
**Suggests** covr, knitr, modeldata, prospectr, rmarkdown, roxygen2, testthat (>= 3.0.0), tidymodels, tidyverse, withr  
**VignetteBuilder** knitr  
**Config/testthat/edition** 3  
**Encoding** UTF-8  
**LazyData** true  
**Roxygen** list(markdown = TRUE)  
**RoxygenNote** 7.3.1  
**Repository** <https://jameshwade.r-universe.dev>  
**RemoteUrl** <https://github.com/JamesHWade/measure>  
**RemoteRef** main  
**RemoteSha** 3234704bef6c442ebb31881fc17fc4251af7e4e2

## Contents

glucose_bioreactors . . . . .	2
meats_long . . . . .	3
required_pkgs.recipe . . . . .	4

step_baseline . . . . .	4
step_measure_input_long . . . . .	5
step_measure_input_wide . . . . .	7
step_measure_output_long . . . . .	9
step_measure_output_wide . . . . .	10
step_measure_savitzky_golay . . . . .	12
subtract_rf_baseline . . . . .	13
tidy.step_measure_input_long . . . . .	14
window_side . . . . .	15

<b>Index</b>	<b>16</b>
--------------	-----------

---

glucose_bioreactors	<i>Raman Spectra Bioreactor Data</i>
---------------------	--------------------------------------

---

**Description**

Kuhn and Johnson (2013) used these two data sets to model the glucose yeild in large- and small-scale bioreactors:

**Details**

- Fifteen small-scale (5 liters) bioreactors were seeded with cells and were monitored daily for 14 days.
- Three large-scale bioreactors were also seeded with cells from the same batch and monitored daily for 14 days.

Samples were collected each day from all bioreactors and glucose was measured. The goal would be to create models on the data from the more numerous small-scale bioreactors and then evaluate if these results can accurately predict what is happening in the large-scale bioreactors.

**Value**

Two tibbles. For each, there are 2,651 columns whose names are numbers and these are the measured assay values (and the names are the wave numbers). The numeric column glucose has the outcome data, day is the number of days in the bioreactor, the batch\_id is the reactor identifier (with "L" for large and "S" for small), and batch\_sample that is the ID and the day.

**Source**

Kuhn and Johnson (2020), *Feature Engineering and Selection*, Chapman and Hall/CRC . <https://bookdown.org/max/FES/> and <https://github.com/topepo/FES>

**Examples**

```
data(glucose_bioreactors)
dim(bioreactors_small)
```

---

meats_long	<i>Fat, water and protein content of meat samples</i>
------------	---

---

## Description

"These data are recorded on a Tecator Infratec Food and Feed Analyzer working in the wavelength range 850 - 1050 nm by the Near Infrared Transmission (NIT) principle. Each sample contains finely chopped pure meat with different moisture, fat and protein contents.

## Details

If results from these data are used in a publication we want you to mention the instrument and company name (Tecator) in the publication. In addition, please send a preprint of your article to

Karin Thente, Tecator AB, Box 70, S-263 21 Hoganas, Sweden

The data are available in the public domain with no responsibility from the original data source. The data can be redistributed as long as this permission note is attached."

"For each meat sample the data consists of a 100 channel spectrum of absorbances and the contents of moisture (water), fat and protein. The absorbance is -log10 of the transmittance measured by the spectrometer. The three contents, measured in percent, are determined by analytic chemistry."

Included here are the meats data transformed to a long format with

```
modeldata::meats %>%  
  rowid_to_column(var = "id") %>%  
  pivot_longer(cols = starts_with("x_"),  
               names_to = "channel",  
               values_to = "transmittance") %>%  
  mutate(channel = str_extract(channel, "[:digit:]+") %>% as.integer())
```

## Value

meats\_long      a tibble

## Examples

```
data(meats_long)  
str(meats_long)
```

---

required_pkgs.recipe	<i>Set package dependencies</i>
----------------------	---------------------------------

---

### Description

Set package dependencies

### Usage

```
## S3 method for class 'step_isomap'
required_pkgs(x, ...)
```

### Arguments

x	A step object.
...	Not used.

---

step_baseline	<i>Fit and subtract a baseline from a measurement signal</i>
---------------	--

---

### Description

Fit and subtract a baseline from a measurement signal

### Usage

```
step_baseline(
  recipe,
  ...,
  role = NA,
  trained = FALSE,
  options = NULL,
  skip = FALSE,
  id = recipes::rand_id("measure")
)
```

### Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose variables for this step.
role	Assign the role of new variables.
trained	A logical to indicate if the quantities for preprocessing have been estimated.

options	A list of options to the default method for <code>stats::prcomp()</code> . Argument defaults are set to <code>retx = FALSE</code> , <code>center = FALSE</code> , <code>scale. = FALSE</code> , and <code>tol = NULL</code> . <b>Note</b> that the argument <code>x</code> should not be passed here (or at all).
skip	A logical. Should the step be skipped when the recipe is baked by <code>bake()</code> ? While all operations are baked when <code>prep()</code> is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using <code>skip = TRUE</code> as it may affect the computations for subsequent operations.
id	A character string that is unique to this step to identify it.

---

step\_measure\_input\_long

*Ingest Measurements from a Single Column*


---

## Description

`step_measure_input_long` creates a *specification* of a recipe step that converts measures organized in a column for the analytical results (and an option column of numeric indices) into an internal format used by the package.

## Usage

```
step_measure_input_long(
  recipe,
  ...,
  location,
  pad = FALSE,
  role = "measure",
  trained = FALSE,
  columns = NULL,
  skip = FALSE,
  id = rand_id("measure_input_long")
)
```

## Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which <i>single</i> column contains the analytical measurements. The selection should be in the order of the measurement's profile.
location	One or more selector functions to choose which <i>single</i> column has the locations of the analytical values.
pad	Whether to pad the measurements to ensure that they all have the same number of values. This is useful when there are missing values in the measurements.

role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
columns	A character vector of column names determined by the recipe.
skip	A logical. Should the step be skipped when the recipe is baked by <code>bake()</code> ? While all operations are baked when <code>prep()</code> is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using <code>skip = TRUE</code> as it may affect the computations for subsequent operations.
id	A character string that is unique to this step to identify it.

## Details

This step is designed for data in a format where there is a column for the analytical measurement (e.g., absorption, etc.) and another with the location of the value (e.g., wave number, etc.).

`step_measure_input_long()` will collect those data and put them into a format used internally by this package. The data structure has a row for each independent experimental unit and a nested tibble with that sample's measure (measurement and location). It assumes that there are unique combinations of the other columns in the data that define individual patterns associated with the pattern. If this is not the case, the special values might be inappropriately restructured.

The best advice is to have a column of any type that indicates the unique sample number for each measure. For example, if there are 200 values in the measure and 7 samples, the input data (in long format) should have 1,400 rows. We advise having a column with 7 unique values indicating which of the rows correspond to each sample.

## Missing Data

Currently, **measure** assumes that there are equal numbers of values within a sample. If there are missing values in the measurements, you'll need to pad them with missing values (as opposed to an absent row in the long format). If not, an error will occur.

## Tidying

When you `tidy()` this step, a tibble indicating which of the original columns were used to reformat the data.

## See Also

Other input/output steps: `step_measure_input_wide()`, `step_measure_output_long()`, `step_measure_output_wide()`

---

step\_measure\_input\_wide

*Ingest Measurements in Separate Columns*


---

## Description

step\_measure\_input\_wide creates a *specification* of a recipe step that converts measures organized in multiple columns into an internal format used by the package.

## Usage

```
step_measure_input_wide(
  recipe,
  ...,
  role = "measure",
  trained = FALSE,
  columns = NULL,
  location_values = NULL,
  skip = FALSE,
  id = rand_id("measure_input_wide")
)
```

## Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose variables for this step. See <a href="#">selections()</a> for more details.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
columns	A character string of the selected variable names. This field is a placeholder and will be populated once <a href="#">prep()</a> is used.
location_values	A numeric vector of values that specify the location of the measurements (e.g., wavelength etc.) in the same order as the variables selected by .... If not specified, a sequence of integers (starting at 1L) is used.
skip	A logical. Should the step be skipped when the recipe is baked by <a href="#">bake()</a> ? While all operations are baked when <a href="#">prep()</a> is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations.
id	A character string that is unique to this step to identify it.

## Details

This step is designed for data in a format where the analytical measurements are in separate columns. `step_measure_input_wide()` will collect those data and put them into a format used internally by this package. The data structure has a row for each independent experimental unit and a nested tibble with that sample's measure (measurement and location). It assumes that there are unique combinations of the other columns in the data that define individual patterns associated with the pattern. If this is not the case, the special values might be inappropriately restructured.

The best advice is to have a column of any type that indicates the unique sample number for each measure. For example, if there are 20 rows in the input data set, the columns that are *not* analytically measurements show have no duplicate combinations in the 20 rows.

## Tidying

When you `tidy()` this step, a tibble indicating which of the original columns were used to reformat the data.

## See Also

Other input/output steps: `step_measure_input_long()`, `step_measure_output_long()`, `step_measure_output_wide()`

## Examples

```
data(meats, package = "modeldata")

# Outcome data is to the right
names(meats) %>% tail(10)

# -----
# Ingest data without adding the location (i.e. wave number) for the spectra

rec <-
  recipe(water + fat + protein ~ ., data = meats) %>%
  step_measure_input_wide(starts_with("x_")) %>%
  prep()

summary(rec)

# -----
# Ingest data without adding the location (i.e. wave number) for the spectra

# Make up some locations for the spectra's x-axis
index <- seq(1, 2, length.out = 100)

rec <-
  recipe(water + fat + protein ~ ., data = meats) %>%
  step_measure_input_wide(starts_with("x_"), location_values = index) %>%
  prep()

summary(rec)
```



---

step\_measure\_output\_long*Reorganize Measurements to Two Columns*

---

## Description

step\_measure\_output\_long creates a *specification* of a recipe step that converts measures to a format with columns for the measurement and the corresponding location (i.e., "long" format).

## Usage

```
step_measure_output_long(  
  recipe,  
  values_to = ".measure",  
  location_to = ".location",  
  role = "predictor",  
  trained = FALSE,  
  skip = FALSE,  
  id = rand_id("measure_output_long")  
)
```

## Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
values_to	A single character string for the column containing the analytical measurement.
location_to	A single character string for the column containing the location of the measurement (e.g. wavenumber or index).
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
skip	A logical. Should the step be skipped when the recipe is baked by <code>bake()</code> ? While all operations are baked when <code>prep()</code> is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using <code>skip = TRUE</code> as it may affect the computations for subsequent operations.
id	A character string that is unique to this step to identify it.

## Details

This step is designed convert analytical measurements from their internal data structure to a two column format.

## See Also

Other input/output steps: [step\\_measure\\_input\\_long\(\)](#), [step\\_measure\\_input\\_wide\(\)](#), [step\\_measure\\_output\\_wide\(\)](#)

**Examples**

```
library(dplyr)

data(glucose_bioreactors)
bioreactors_small$batch_sample <- NULL

small_tr <- bioreactors_small[1:200, ]
small_te <- bioreactors_small[201:210, ]

small_rec <-
  recipe(glucose ~ ., data = small_tr) %>%
  update_role(batch_id, day, new_role = "id columns") %>%
  step_measure_input_wide(`400`:`3050`) %>%
  prep()

# Before reformatting:

small_rec %>% bake(new_data = small_te)

# After reformatting:

output_rec <-
  small_rec %>%
  step_measure_output_long() %>%
  prep()

output_rec %>% bake(new_data = small_te)
```

---

step\_measure\_output\_wide

*Reorganize Measurements to Separate Columns*


---

**Description**

step\_measure\_output\_wide creates a *specification* of a recipe step that converts measures to multiple columns (i.e., "wide" format).

**Usage**

```
step_measure_output_wide(
  recipe,
  prefix = "measure_",
  role = "predictor",
  trained = FALSE,
  skip = FALSE,
  id = rand_id("measure_output_wide")
)
```

**Arguments**

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
prefix	A character string used to name the new columns.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
skip	A logical. Should the step be skipped when the recipe is baked by <code>bake()</code> ? While all operations are baked when <code>prep()</code> is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using <code>skip = TRUE</code> as it may affect the computations for subsequent operations.
id	A character string that is unique to this step to identify it.

**Details**

This step is designed convert analytical measurements from their internal data structure to separate columns.

Wide outputs can be helpful when you want to use standard recipes steps with the measurements, such as `recipes::step_pca()`, `recipes::step_pls()`, and so on.

**See Also**

Other input/output steps: `step_measure_input_long()`, `step_measure_input_wide()`, `step_measure_output_long()`

**Examples**

```
library(dplyr)

data(glucose_bioreactors)
bioreactors_small$batch_sample <- NULL

small_tr <- bioreactors_small[1:200, ]
small_te <- bioreactors_small[201:210, ]

small_rec <-
  recipe(glucose ~ ., data = small_tr) %>%
  update_role(batch_id, day, new_role = "id columns") %>%
  step_measure_input_wide(`400`:`3050`) %>%
  prep()

# Before reformatting:

small_rec %>% bake(new_data = small_te)

# After reformatting:

output_rec <-
  small_rec %>%
  step_measure_output_wide() %>%
```

```

prep()

output_rec %>% bake(new_data = small_te)

```

---

```

step_measure_savitzky_golay
      Savitzky-Golay Pre-Processing

```

---

## Description

`step_measure_savitzky_golay` creates a *specification* of a recipe step that smooths and filters the measurement sequence.

## Usage

```

step_measure_savitzky_golay(
  recipe,
  role = NA,
  trained = FALSE,
  degree = 3,
  window_side = 11,
  differentiation_order = 0,
  skip = FALSE,
  id = rand_id("measure_savitzky_golay")
)

```

## Arguments

<code>recipe</code>	A recipe object. The step will be added to the sequence of operations for this recipe.
<code>role</code>	Not used by this step since no new variables are created.
<code>trained</code>	A logical to indicate if the quantities for preprocessing have been estimated.
<code>degree</code>	An integer for the polynomial degree to use for smoothing.
<code>window_side</code>	An integer for how many units there are on each side of the window. This means that <code>window_side = 1</code> has a total window width of 3 (e.g., width is $2 * window\_side + 1$ ).
<code>differentiation_order</code>	An integer for the degree of filtering (zero indicates no differentiation).
<code>skip</code>	A logical. Should the step be skipped when the recipe is baked by <code>bake()</code> ? While all operations are baked when <code>prep()</code> is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using <code>skip = TRUE</code> as it may affect the computations for subsequent operations
<code>id</code>	A character string that is unique to this step to identify it.

## Details

This method can both smooth out random noise and reduce between-predictor correlation. It fits a polynomial to a window of measurements and this results in fewer measurements than the input. Measurements are assumed to be equally spaced.

The polynomial degree should be less than the window size. Also, window size must be greater than polynomial degree. If either case is true, the original argument values are increased to satisfy these conditions (with a warning).

**No selectors should be supplied to this step function.** The data should be in a special internal format produced by `step_measure_input_wide()` or `step_measure_input_long()`.

The measurement locations are reset to integer indices starting at one.

## Value

An updated version of recipe with the new step added to the sequence of any existing operations.

## Tidying

When you `tidy()` this step, a tibble with columns is returned.

## Examples

```
if (rlang::is_installed("prospectr")) {
  rec <-
    recipe(water + fat + protein ~ ., data = meats_long) %>%
    update_role(id, new_role = "id") %>%
    step_measure_input_long(transmittance, location = vars(channel)) %>%
    step_measure_savitzky_golay(
      differentiation_order = 1,
      degree = 3,
      window_side = 5
    ) %>%
    prep()
}
```

---

subtract_rf_baseline	<i>Subtract baseline using robust fitting method</i>
----------------------	--

---

## Description

Subtract baseline using robust fitting method

## Usage

```
subtract_rf_baseline(data, yvar, span = 2/3, maxit = c(5, 5))
```

**Arguments**

<code>data</code>	A dataframe containing the variable for baseline subtraction
<code>yvar</code>	The name of the column for baseline subtraction
<code>span</code>	Controls the amount of smoothing based on the fraction of data to use in computing each fitted value, defaults to 2/3.
<code>maxit</code>	The number of iterations to use the robust fit, defaults to <code>c(5, 5)</code> where the first value specifies iterations for asymmetric weighting function and the second value for symmetric weighting function.

**Value**

A dataframe matching column in data plus raw and baseline columns

**Examples**

```
meats_long %>% subtract_rf_baseline(yvar = transmittance)
```

---

```
tidy.step_measure_input_long
```

*Tidiers for measure steps*

---

**Description**

Tidiers for measure steps

**Usage**

```
## S3 method for class 'step_measure_input_long'
tidy(x, ...)

## S3 method for class 'step_measure_input_wide'
tidy(x, ...)

## S3 method for class 'step_measure_output_long'
tidy(x, ...)

## S3 method for class 'step_measure_output_wide'
tidy(x, ...)

## S3 method for class 'step_measure_savitzky_golay'
tidy(x, ...)
```

**Arguments**

<code>x</code>	A step object.
<code>...</code>	Not used.

---

window_side	<i>Parameter for measure steps</i>
-------------	------------------------------------

---

### Description

window\_side() and differentiation\_order() are used with Savitzky-Golay processing.

### Usage

```
window_side(range = c(1L, 5L), trans = NULL)
```

```
differentiation_order(range = c(0L, 4L), trans = NULL)
```

### Arguments

range	A two-element vector holding the <i>defaults</i> for the smallest and largest possible values, respectively. If a transformation is specified, these values should be in the <i>transformed units</i> .
trans	A trans object from the scales package, such as scales::transform_log10() or scales::transform_reciprocal(). If not provided, the default is used which matches the units used in range. If no transformation, NULL.

### Details

This parameter is often used to correct for zero-count data in tables or proportions.

### Value

A function with classes "quant\_param" and "param".

### Examples

```
window_side()
differentiation_order()
```

# Index

- \* **datasets**
  - glucose\_bioreactors, [2](#)
  - meats\_long, [3](#)
- \* **input/output steps**
  - step\_measure\_input\_long, [5](#)
  - step\_measure\_input\_wide, [7](#)
  - step\_measure\_output\_long, [9](#)
  - step\_measure\_output\_wide, [10](#)
- \* **measure-differencing**
  - step\_measure\_savitzky\_golay, [12](#)
- \* **measure-smoothing**
  - step\_measure\_savitzky\_golay, [12](#)
- bake(), [5–7](#), [9](#), [11](#), [12](#)
- bioreactors\_large
  - (glucose\_bioreactors), [2](#)
- bioreactors\_small
  - (glucose\_bioreactors), [2](#)
- differentiation\_order (window\_side), [15](#)
- glucose\_bioreactors, [2](#)
- meats\_long, [3](#)
- prep(), [5–7](#), [9](#), [11](#), [12](#)
- recipes::step\_pca(), [11](#)
- recipes::step\_pls(), [11](#)
- required\_pkgs.recipe, [4](#)
- required\_pkgs.step\_isomap
  - (required\_pkgs.recipe), [4](#)
- selections(), [7](#)
- stats::prcomp(), [5](#)
- step\_baseline, [4](#)
- step\_measure\_input\_long, [5](#), [8](#), [9](#), [11](#)
- step\_measure\_input\_long(), [13](#)
- step\_measure\_input\_wide, [6](#), [7](#), [9](#), [11](#)
- step\_measure\_input\_wide(), [13](#)
- step\_measure\_output\_long, [6](#), [8](#), [9](#), [11](#)
- step\_measure\_output\_wide, [6](#), [8](#), [9](#), [10](#)
- step\_measure\_savitzky\_golay, [12](#)
- subtract\_rf\_baseline, [13](#)
- tidy(), [6](#), [8](#), [13](#)
- tidy.step\_measure\_input\_long, [14](#)
- tidy.step\_measure\_input\_wide
  - (tidy.step\_measure\_input\_long), [14](#)
- tidy.step\_measure\_output\_long
  - (tidy.step\_measure\_input\_long), [14](#)
- tidy.step\_measure\_output\_wide
  - (tidy.step\_measure\_input\_long), [14](#)
- tidy.step\_measure\_savitzky\_golay
  - (tidy.step\_measure\_input\_long), [14](#)
- window\_side, [15](#)