

# Package: studentlife (via r-universe)

October 26, 2024

**URL** <https://github.com/Frycast/studentlife>

**BugReports** <https://github.com/Frycast/studentlife/issues>

**Type** Package

**Title** Tidy Handling and Navigation of the Student-Life Dataset

**Version** 1.1.0

**Description** Download, navigate and analyse the Student-Life dataset. The Student-Life dataset contains passive and automatic sensing data from the phones of a class of 48 Dartmouth college students. It was collected over a 10 week term. Additionally, the dataset contains ecological momentary assessment results along with pre-study and post-study mental health surveys. The intended use is to assess mental health, academic performance and behavioral trends. The raw dataset and additional information is available at <https://studentlife.cs.dartmouth.edu/>.

**Depends** R (>= 3.4.0)

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Imports** purrr (>= 0.3.2), readr (>= 1.3.1), tidyr (>= 0.8.3), dplyr (>= 0.8.0.1), jsonlite (>= 1.6), tibble (>= 2.0.1), R.utils (>= 2.8.0), skimr (>= 1.0.7), visdat (>= 0.5.3), ggplot2 (>= 3.1.1), crayon (>= 1.3.4)

**RoxygenNote** 7.1.1

**Suggests** testthat

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

**RemoteUrl** <https://github.com/frycast/studentlife>

**RemoteRef** HEAD

**RemoteSha** efd968d10a68b0f1b1f08a696da3d27f3a1a797

## Contents

add_block_labels	2
download_studentlife	3
get_EMA_questions	5
get_schema	5
get_table	6
is_dateless_SL_tibble	7
is_dateonly_SL_tibble	7
is_interval_SL_tibble	8
is_reg_SL_tibble	9
is_SL_tibble	9
is_timestamp_SL_tibble	10
load_SL_tibble	11
PAM_categorise	13
regularise_time	14
response_hour_hist	15
SL_tables	16
studentlife	17
vis_NAs	17
vis_response_counts	18

<b>Index</b>	<b>20</b>
--------------	-----------

---

add_block_labels	<i>add_block_labels</i>
------------------	-------------------------

---

## Description

Classify observations from an `SL_tibble` into block labels using available date-time information. See more information about "blocks" under the details section. Daylight savings is ignored, and started on 31st March 2013.

## Usage

```
add_block_labels(
  tab,
  type = c("hour_in_day", "epoch", "day", "week", "weekday", "month", "date"),
  interval = "start",
  warning = TRUE,
  start_date = getOption("SL_start"),
  epoch_levels = getOption("SL_epoch_levels"),
  epoch_ubs = getOption("SL_epoch_ubs"),
  unsafe = F
)
```

**Arguments**

tab	An SL_tibble as returned by the function <code>load_SL_tibble</code> .
type	A character vector of block label types to include. Can be one or more of "epoch", "day", "week", "weekday", "month" and "date". Any block label types that are not inferable from the available date-time data are ignored.
interval	A character string that decides how block membership is decided when tab is of class <code>interval_SL_tibble</code> . Can be either "start" (use <code>start_timestamp</code> ), "end" (use <code>end_timestamp</code> ) or "middle" (use the midpoint between <code>start_timestamp</code> and <code>end_timestamp</code> ).
warning	Logical. If TRUE then a warning is produced whenever a block label type is not inferable from the available date-time data.
start_date	Date. The date that the StudentLife study started.
epoch_levels	A character vector of epoch levels.
epoch_ubs	An integer vector that defines the hour that is the upper boundary of each epoch.
unsafe	A logical. Default is FALSE. If this is set to TRUE then less checks will be performed.

**Details**

Block label types can be one or more of "epoch" (giving labels morning, evening, afternoon and night), "day" (giving number of days since the `start_date` of the StudentLife study), "week" (giving integer number of weeks since the first week of the StudentLife study, rounded downs), "weekday" (giving the day of the week), "month" (giving integer number of months since the start of the StudentLife study, rounded down) and "date".

**Examples**

```
d <- tempdir()
download_studentlife(location = d, url = "testdata")

tab <- load_SL_tibble(
  loc = d, schema = "sensing", table = "activity", csv_nrows = 10)

b_tab <- add_block_labels(tab)
b_tab
```

---

download\_studentlife    *download\_studentlife*

---

**Description**

Download the entire StudentLife dataset or a smaller sample dataset for testing.

## Usage

```
download_studentlife(  
  url = "dartmouth",  
  location = ".",  
  unzip = TRUE,  
  untar = TRUE  
)
```

## Arguments

url	A character string. Either "rdata" for the URL to the (more efficient) RData format version hosted on Zenodo, or "dartmouth" for the (original) Dartmouth URL, or "testdata" for a small sample dataset. Otherwise a full URL of your choice can be specified leading to the StudentLife dataset as a .tar.gz file.
location	The destination path. If the path does not exist it is created with <code>dir.create</code>
unzip	Logical. If TRUE then the dataset will be unzipped with <code>bunzip2</code> . Leave as default unless you plan to do it manually.
untar	Logical. If TRUE then the dataset will be untarred with <code>untar</code> . Leave as default unless you plan to do it manually.

## Details

If url = "rdata" then data will be downloaded from <https://zenodo.org/record/3529253> If url = "dartmouth" then data will be downloaded from <https://studentlife.cs.dartmouth.edu/dataset/dataset.tar.bz2> If url = "testdata" then data will be downloaded from the test data at the studentlife GitHub repository <https://github.com/frycast/studentlife>

## Examples

```
d <- tempdir()  
download_studentlife(location = d, url = "testdata")  
  
## Not run:  
## With menu  
load_SL_tibble(location = d)  
  
## End(Not run)  
  
## Without menu  
SL_tables  
load_SL_tibble(schema = "EMA", table = "PAM", location = d)
```

---

get_EMA_questions	<i>get_EMA_questions</i>
-------------------	--------------------------

---

**Description**

Get the EMA questions from a StudentLife tibble whose schema is "EMA".

**Usage**

```
get_EMA_questions(x)
```

**Arguments**

x                    A StudentLife tibble whose schema is EMA, as output by the function [load\\_SL\\_tibble](#).

**Value**

The EMA\_questions attribute of x

**Examples**

```
d <- tempdir()
download_studentlife(location = d, url = "testdata")

tab_PAM <- load_SL_tibble(schema = "EMA", table = "PAM", location = d)

# Returns "PAM"
get_EMA_questions(tab_PAM)
```

---

get_schema	<i>get_schema</i>
------------	-------------------

---

**Description**

Retrieve the schema name from a StudentLife tibble

**Usage**

```
get_schema(x)
```

**Arguments**

x                    An object of class StudentLife tibble (SL\_tbl), as produced by the function [load\\_SL\\_tibble](#).

**Value**

A character string indicating the schema name

**Examples**

```
d <- tempdir()
download_studentlife(location = d, url = "testdata")

tab_PAM <- load_SL_tibble(schema = "EMA", table = "PAM", location = d)

# Returns "EMA"
get_schema(tab_PAM)
```

---

get\_table

*get\_table*

---

**Description**

Retrieve the table name from a StudentLife tibble

**Usage**

```
get_table(x)
```

**Arguments**

x An object of class StudentLife tibble (SL\_tbl), as produced by the function [load\\_SL\\_tibble](#).

**Value**

A character string indicating the table name

**Examples**

```
d <- tempdir()
download_studentlife(location = d, url = "testdata")

tab_PAM <- load_SL_tibble(schema = "EMA", table = "PAM", location = d)

# Returns "PAM"
get_table(tab_PAM)
```

---

*is\_dateless\_SL\_tibble* *is\_dateless\_SL\_tibble*

---

### **Description**

Confirm that an object is a dateless StudentLife tibble

### **Usage**

```
is_dateless_SL_tibble(x)
```

### **Arguments**

x                    Any object

### **Value**

Logical

### **Examples**

```
d <- tempdir()
download_studentlife(location = d, url = "testdata")

tab_S <- load_SL_tibble(
  schema = "survey", table = "BigFive", location = d)

# Returns TRUE
is_dateless_SL_tibble(tab_S)
```

---

*is\_dateonly\_SL\_tibble* *is\_dateonly\_SL\_tibble*

---

### **Description**

Confirm that an object is a date-only StudentLife tibble

### **Usage**

```
is_dateonly_SL_tibble(x)
```

### **Arguments**

x                    Any object

**Value**

Logical

**Examples**

```
d <- tempdir()
download_studentlife(location = d, url = "testdata")

tab_DL <- load_SL_tibble(
  schema = "education", table = "deadlines", location = d)

# Returns TRUE
is_dateonly_SL_tibble(tab_DL)
```

---

`is_interval_SL_tibble` *is\_interval\_SL\_tibble*

---

**Description**

Confirm that an object is an interval StudentLife tibble

**Usage**

```
is_interval_SL_tibble(x)
```

**Arguments**

x                    Any object

**Value**

Logical

**Examples**

```
d <- tempdir()
download_studentlife(location = d, url = "testdata")

tab_con <- load_SL_tibble(
  schema = "sensing", table = "conversation", location = d, csv_nrow = 10)

# Returns TRUE
is_interval_SL_tibble(tab_con)
```



---

is_reg_SL_tibble	<i>is_reg_SL_tibble</i>
------------------	-------------------------

---

**Description**

Confirm that an object is a regularised StudentLife tibble

**Usage**

```
is_reg_SL_tibble(x)
```

**Arguments**

x                    Any object

**Value**

Logical

**Examples**

```
d <- tempdir()
download_studentlife(location = d, url = "testdata")

tab_PAM <- load_SL_tibble(schema = "EMA", table = "PAM", location = d)

reg_PAM <- regularise_time(
  tab_PAM, blocks = c("day", "epoch"), m = mean(picture_idx, na.rm = TRUE))

# Returns TRUE
is_reg_SL_tibble(reg_PAM)
```

---

is_SL_tibble	<i>is_SL_tibble</i>
--------------	---------------------

---

**Description**

Confirm that an object is a StudentLife tibble

**Usage**

```
is_SL_tibble(x)
```

**Arguments**

x                    Any object

**Value**

Logical

**Examples**

```
d <- tempdir()
download_studentlife(location = d, url = "testdata")

tab_PAM <- load_SL_tibble(schema = "EMA", table = "PAM", location = d)

# Returns TRUE
is_SL_tibble(tab_PAM)
```

---

`is_timestamp_SL_tibble`

*is\_timestamp\_SL\_tibble*

---

**Description**

Confirm that an object is a timestamped StudentLife tibble

**Usage**

```
is_timestamp_SL_tibble(x)
```

**Arguments**

x                    Any object

**Value**

Logical

**Examples**

```
d <- tempdir()
download_studentlife(location = d, url = "testdata")

tab_PAM <- load_SL_tibble(schema = "EMA", table = "PAM", location = d)

# Returns TRUE
is_timestamp_SL_tibble(tab_PAM)
```

---

load_SL_tibble	<i>load_SL_tibble</i>
----------------	-----------------------

---

### Description

Import a chosen StudentLife table as a tibble. Leave schema and table unspecified to choose interactively via a menu. This function is only intended for use with the studentlife dataset in its original format, with the original directory structure. See the examples below for the recommended alternative approach to loading tables when the RData format is used.

### Usage

```
load_SL_tibble(
  schema,
  table,
  location = ".",
  time_options = c("interval", "timestamp", "dateonly", "dateless"),
  vars,
  csv_nrows,
  datafolder = "dataset",
  uid_range = getOption("SL_uids")
)
```

### Arguments

schema	A character string. The menu 1 choice. Leave blank to choose interactively.
table	A character string. The menu 2 choice. Leave blank to choose interactively.
location	The path to a copy of the StudentLife dataset.
time_options	A character vector specifying which table types (out of "interval", "timestamp", "dateonly" and "dateless") to include in the menu. This allows you to restrict menu options according to the amount of date-time information present in the data. The default includes all data. Note this parameter only has an effect when used with the interactive menu.
vars	Character vector of variable names to import for all students. Leave blank and this will be chosen interactively if necessary. If vars contains "timestamp" then effort will be made to convert "timestamp" to appropriate variable name(s) for the target table.
csv_nrows	An integer specifying the number of rows to read per student if the target is a csv. The largest files in StudentLife are csv files, so this allows code testing with less overhead.
datafolder	Specifies the subfolder of location that contains the relevant data. This should normally be left as the default.
uid_range	An integer vector. The range of uids in the StudentLife study.

**Value**

An object of class `SL_tibble` is returned. These inherit properties from class `tibble` and class `data.frame`. Depending on the date-time information available, the object may also be a `timestamp_SL_tibble`, `interval_SL_tibble` or `dateonly_SL_tibble` (which are all subclasses of `SL_tibble`).

**Examples**

```
## Example that uses RData format to efficiently
## download and load tables, as an alternative
## to using this function.
## Not run:
d <- tempdir()
download_studentlife(location = d, url = "rdata")

# Choose the schema and table from the list SL_tables:
SL_tables

# Example with activity table from sensing schema
schema <- "sensing"
table <- "activity"
act <- readRDS(paste0(d, "/dataset_rds/", schema, "/", table, ".Rds"))
act

## End(Not run)

## Example that uses the studentlife dataset in
## its original format.

# Use url = "dartmouth" for the full original dataset
d <- tempdir()
download_studentlife(location = d, url = "testdata")

## Not run:
## With menu
load_SL_tibble(location = d)

## End(Not run)

## Without menu
SL_tables
PAM <- load_SL_tibble(schema = "EMA", table = "PAM", location = d)

## Load less data for testing with less overhead
act <- load_SL_tibble(schema = "sensing", table = "activity",
                     location = d, csv_nrows = 10)

## Not run:
## Browse all tables with timestamps (non-interval)
load_SL_tibble(location = d, time_options = "timestamp")

## Browse all tables with intervals
load_SL_tibble(location = d, time_options = "interval")
```

```
## Browse all dateless tables
load_SL_tibble(location = d, time_options = "dateless")

## End(Not run)
```

---

PAM_categorise	<i>PAM_categorise</i>
----------------	-----------------------

---

### Description

Categorise Photographic Affect Meter (PAM) scores into 4 categories by either PAM Quadrant, Valence or Arousal (or multiple of these).

### Usage

```
PAM_categorise(
  tab,
  pam_name = "picture_idx",
  types = c("quadrant", "valence", "arousal")
)
```

### Arguments

tab	A data.frame (or tibble) with a column representing Photographic Affect Meter (PAM) score.
pam_name	Character. The name of the column representing PAM.
types	Character vector containing the categories, one or more of "quadrant", "valence" and "arousal" into which to code PAM scores.

### Details

The 4 Quadrant categories are as follows: Quadrant 1: negative valence, low arousal. Quadrant 2: negative valence, high arousal. Quadrant 3: positive valence, low arousal. Quadrant 4: positive valence, high arousal.

Valence and arousal are traditionally scores from -2 to 2, measuring displeasure to pleasure, and state of activation respectively. However, here we map those scores to positive numbers so (-2,-1,1,2) -> (1,2,3,4).

### Value

The data.frame (or tibble) tab with extra columns pam\_q, pam\_v, and pam\_a for quadrant, valence and arousal respectively.

### References

Pollak, J. P., Adams, P., & Gay, G. (2011, May). PAM: a photographic affect meter for frequent, in situ measurement of affect. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 725-734). ACM.

**Examples**

```
d <- tempdir()
download_studentlife(location = d, url = "testdata")

tab <- load_SL_tibble(
  loc = d, schema = "EMA", table = "PAM", csv_nrows = 10)

PAM_categorise(tab)
```

---

regularise_time	<i>regularise_time</i>
-----------------	------------------------

---

**Description**

Transform an `SL_tibble` (as produced by `load_SL_tibble`) in such a way that the observations are aggregated in equal length intervals called 'blocks' (for more information on blocks see [add\\_block\\_labels](#)).

**Usage**

```
regularise_time(
  tab,
  ...,
  blocks = c("epoch", "day"),
  add_NAs = TRUE,
  unsafe = F,
  study_duration = getOption("SL_duration"),
  start_date = getOption("SL_start"),
  epoch_levels = getOption("SL_epoch_levels"),
  epoch_ubs = getOption("SL_epoch_ubs"),
  uid_range = getOption("SL_uids"),
  date_range = seq(from = start_date, by = 1, length.out = study_duration)
)
```

**Arguments**

tab	An <code>SL_tibble</code> as returned by the function <code>load_SL_tibble</code> . The <code>SL_tibble</code> must have some date-time information.
...	Arguments passed to <code>summarise</code> , used to aggregate values when multiple observations are encountered in a block. Any columns not specified here or under <code>blocks</code> will be dropped.
blocks	A character vector naming one or more of the block options "hour_in_day", "epoch", "day", "week", "weekday", "month" or "date". If not present as column names in <code>tab</code> , an attempt will be made to infer the blocks from existing time information with <code>add_block_labels</code> . The returned data frame will have one observation (possibly NA) for each block.

add_NAs	A logical. If TRUE then NAs will be introduced to fill missing blocks.
unsafe	A logical. Default is FALSE. If this is set to TRUE then less checks will be performed.
study_duration	Integer. The duration of the StudentLife study in days. This parameter does nothing if <code>limit_date_range</code> is TRUE.
start_date	Date. The date that the StudentLife study started.
epoch_levels	A character vector of epoch labels.
epoch_ubs	An integer vector that defines the hour that is the upper boundary of each epoch.
uid_range	An integer vector. The range of uids in the StudentLife study.
date_range	A vector of dates to be used if <code>limit_date_range</code> is FALSE.

### Examples

```
d <- tempdir()
download_studentlife(location = d, url = "testdata")

tab <- load_SL_tibble(
  loc = d, schema = "sensing", table = "activity", csv_nrows = 10)

r_tab <- regularise_time(
  tab, blocks = c("day", "weekday"),
  act_inf = max(activity_inference), add_NAs = FALSE)

r_tab
```

---

response\_hour\_hist      *response\_hour\_hist*

---

### Description

This function produces a histogram that visualizes the frequencies of observations within hourly blocks, or blocks of multiple hours.

### Usage

```
response_hour_hist(
  tab,
  break_hours = 10,
  xlab = "Hours into study",
  main = paste0("Distribution of ", attr(tab, "table"), " response times"),
  ...
)
```

**Arguments**

tab	A StudentLife tibble with time information, (i.e., and object of class <code>timestamp_SL_tbl</code> or <code>interval_SL_tbl</code> ) as can be returned by the function <code>load_SL_tibble</code> .
break_hours	Specify the width in hours of each histogram bin.
xlab	Argument passed to <code>hist</code> .
main	Argument passed to <code>hist</code> .
...	Arguments passed to <code>hist</code> .

**Examples**

```
d <- tempdir()
download_studentlife(location = d, url = "testdata")

tab_PAM <- load_SL_tibble(schema = "EMA", table = "PAM", location = d)

response_hour_hist(tab_PAM)
```

---

SL\_tables

*List of all the tables available in the StudentLife dataset.*

---

**Description**

This command returns a 5 element list. Each of the five elements are given names corresponding to the schema names of the studentlife data set. Each element is a vector of strings, where each string corresponds to the name of a table within the respective schema.

**Usage**

```
SL_tables
```

**Format**

An object of class `list` of length 5.

**Source**

<https://studentlife.cs.dartmouth.edu/>



## Description

Download, navigate and analyse the Student-Life dataset. The Student-Life dataset contains passive and automatic sensing data from the phones of a class of 48 de-identified Dartmouth college students. It was collected over a 10 week term. Additionally, the dataset contains Ecological Momentary Assessment results along with pre- and post-study mental health surveys, such as the PHQ-9. The intended use is to assess mental health, academic performance and behavioral trends. The raw dataset and additional information is available at <<https://studentlife.cs.dartmouth.edu/>>.

## Details

Details on the Student-Life dataset as well as the dataset itself are available at <https://studentlife.cs.dartmouth.edu/>.

## Update

Current updates are available through URL: <https://github.com/frycast/studentlife>

## BugReports

<https://github.com/frycast/studentlife/issues>

## Author(s)

Daniel Fryer <[d.fryer@latrobe.edu.au](mailto:d.fryer@latrobe.edu.au)>

---

vis\_NAs*vis\_NAs*

---

## Description

Produce a visualisation of the number of missing values among each student in a regularised SL\_tbl.

## Usage

```
vis_NAs(  
  tab,  
  response,  
  main = paste0("Missing values by student (", attr(tab, "table"), ") (blocks: ",  
    paste0(attr(tab, "blocks"), collapse = ", "), ")"),  
  show_perc_col = FALSE,  
  ...  
)
```

**Arguments**

tab	A regularised StudentLife tibble (i.e., an object of class <code>reg_SL_tbl</code> ) as produced by the function <code>regularise_time</code> .
response	A character string naming one of the columns in <code>tab</code> that is not in <code>attr(tab, "blocks")</code> . If missing then this defaults to the first such column name.
main	The plot title, passed to <code>ggtitle</code> .
show_perc_col	Logical passed to <code>vis_miss</code> . TRUE adds in the percentage of missing data in each column into the x axis.
...	Arguments passed to <code>vis_miss</code> .

**Value**

A ggplot object.

**Examples**

```
d <- tempdir()
download_studentlife(location = d, url = "testdata")

tab_PAM <- load_SL_tibble(schema = "EMA", table = "PAM", location = d)

reg_PAM <- regularise_time(
  tab_PAM, blocks = c("day", "epoch"), m = mean(picture_idx, na.rm = TRUE))

vis_NAs(reg_PAM, response = "m")
```

---

vis\_response\_counts    *vis\_response\_counts*

---

**Description**

Produce an ordered bar plot of the total number of responses for each student in a regularised `SL_tbl`.

**Usage**

```
vis_response_counts(
  tab,
  response,
  main = paste0("Total responses by student (", attr(tab, "table"), ")"),
  xlab = "Student UID",
  ylab = "Response count",
  ...
)
```

**Arguments**

tab	A regularised StudentLife tibble (i.e., an object of class <code>reg_SL_tbl</code> ) as produced by the function <code>regularise_time</code> .
response	A character string naming one of the columns in <code>tab</code> that is not in <code>attr(tab, "blocks")</code> . If missing then this defaults to the first such column name.
main	The plot title, passed to <code>barplot</code> .
xlab	The x axis label, passed to <code>barplot</code> .
ylab	The y axis label, passed to <code>barplot</code> .
...	Arguments passed to <code>barplot</code> .

**Value**

A named numeric vector of response counts, sorted in descending order.

**Examples**

```
d <- tempdir()
download_studentlife(location = d, url = "testdata")

tab_PAM <- load_SL_tibble(schema = "EMA", table = "PAM", location = d)

reg_PAM <- regularise_time(
  tab_PAM, blocks = c("day", "epoch"), m = mean(picture_idx, na.rm = TRUE))

vis_response_counts(reg_PAM, response = "m")
```

# Index

## \* datasets

- SL\_tables, 16
- add\_block\_labels, 2, 14
- barplot, 19
- bunzip2, 4
- data.frame, 12
- dir.create, 4
- download\_studentlife, 3
- get\_EMA\_questions, 5
- get\_schema, 5
- get\_table, 6
- ggtitle, 18
- hist, 16
- is\_dateless\_SL\_tibble, 7
- is\_dateonly\_SL\_tibble, 7
- is\_interval\_SL\_tibble, 8
- is\_reg\_SL\_tibble, 9
- is\_SL\_tibble, 9
- is\_timestamp\_SL\_tibble, 10
- load\_SL\_tibble, 3, 5, 6, 11, 14, 16
- PAM\_categorise, 13
- regularise\_time, 14, 18, 19
- response\_hour\_hist, 15
- SL\_tables, 16
- studentlife, 17
- studentlife-package (studentlife), 17
- summarise, 14
- tibble, 12
- untar, 4
- vis\_miss, 18
- vis\_NAs, 17
- vis\_response\_counts, 18