Package 'adheRenceRX'

May 7, 2026

Type Package

Title Assess Medication Adherence from Pharmaceutical Claims Data

Version 1.0.0

Date 2020-10-27

Description A (mildly) opinionated set of functions to help assess medication adherence for researchers working with medication claims data.
Medication adherence analyses have several complex steps that are often convoluted and can be time-intensive. The focus is to create a
set of functions using ``tidy principles" geared towards transparency, speed, and flexibility while working with adherence metrics. All functions perform exactly one task
with an intuitive name so that a researcher can handle details (often achieved with vectorized solutions) while we handle non-vectorized tasks common to most
adherence calculations such as adjusting fill dates and determining episodes of care. The methodologies in referenced in this package come from
Canfield SL, et al (2019) ``Navigating the Wild West of Medication Adherence Reporting in Specialty Pharmacy" <doi:10.18553/jmcp.2019.25.10.1073>.

License GPL (>= 2)

Imports Rcpp (>= 1.0.5), anytime, tidyr, dplyr, purrr, lubridate, rlang

LinkingTo Rcpp

RoxygenNote 7.1.0

Depends R (>= 2.10)

LazyData true

Suggests testthat, spelling

URL https://github.com/btbeal/adheRenceRX

BugReports https://github.com/btbeal/adheRenceRX/issues

Language en-US

NeedsCompilation yes

Author Brennan Beal [aut, cre]

Maintainer Brennan Beal <brennanbeal@gmail.com>

Repository CRAN

Date/Publication 2020-11-20 10:40:10 UTC

Contents

`big_data_toy` *A Toy Claims* `tibble` *for Performance Demonstration*

Description

A toy pharmaceutical claims data set meant to be used to benchmark other algorithms

Usage

```
big_data_toy
```

Format

A `tibble` with 100,000 rows and 6 variables:

ID patient ID to be grouped by

date date of claim

days_supply number of days supplied

`calculate_pdc` *Calculate Proportion Days Covered*

Description

Calculate the proportion of days covered (PDC) from a pharmaceutical claims database. This function is suggested only after one has properly adjusted their dates (`propagate_date()`) and identified gaps in therapy (`identify_gaps()`). This function calculates a length of total therapy as the first fill date to the last for a given grouping. Finally, if you'd like to view adherence by episodes after you have used `rank_episodes()`, the function will re-adjust gaps for you so that the gap that defined the episode isn't included.

Usage

```
calculate_pdc(.data, .summarise = TRUE)
```

Arguments

`.data`	data frame
`.summarise`	Logical value (defaulting to TRUE) indicating whether the output should be summarised or not

Value

a summarised tibble, by default, with proportion of days covered calculated

Examples

```
library(adheRenceRX)
library(dplyr)

toy_claims %>%
  group_by(ID) %>%
  propagate_date(.date = date, .days_supply = days_supply) %>%
  identify_gaps() %>%
  calculate_pdc()

#OR, one could group by the ID and episode of care like...
toy_claims %>%
  group_by(ID) %>%
  propagate_date(.date = date, .days_supply = days_supply) %>%
  identify_gaps() %>%
  rank_episodes(.permissible_gap = 30) %>%
  ungroup() %>%
  group_by(ID, episode) %>%
  calculate_pdc()
```

date_check *Restructuring Dates to Remove Overlap*

Description

This is a function meant to be utilized within propagate_date() in order to adjust pharmaceutical claims dates to prevent overlapping in adherence calculations, per Canfield SL, Zuckerman A, Anguiano RH, Jolly JA, DeClercq J. Navigating the wild west of medication adherence reporting in specialty pharmacy. J Manag Care Spec Pharm. 2019;25(10):1073-77.

Usage

```
date_check(df)
```

Arguments

df	a claims data frame with a date of a medication claim and the corresponding days supply

Value

A new claims data frame with an appended column, `adjusted_date`

episode_check *Ranking Episodes of Care*

Description

This is a helper function to assist `rank_episodes`

Usage

```
episode_check(df)
```

Arguments

df	a data frame with "gap", "initial_rank", and "permi_gap" columns appended from `identify_gaps()`

Value

a data frame with an "episode" column appended, which ranks episodes of care in time

`identify_gaps` *Identify Gaps in Therapy*

Description

Compute gaps in a patient's therapy from the end of their prior fill to the beginning of the next. This function assumes that one has arranged the dates and grouped appropriately outside of the function. The length of any gap will be appended to the row after the gap has occurred.

Usage

```
identify_gaps(.data)
```

Arguments

`.data`	data frame

Value

A new claims tibble with an appended column, `gap`

Note

This function relies an `adjusted_date` column to identify gaps in therapy. So, if you don't want to use `propagate_date()` beforehand, you'll need to rename the date variable you wish to use to `adjusted_date`.

Examples

```
library(adheRenceRX)
library(dplyr)

toy_claims %>%
  filter(ID == "D") %>%
  propagate_date(.date_var = date, .days_supply_var = days_supply) %>%
  identify_gaps()
```

`propagate_date` *Adjust Overlapping Fill Dates*

Description

When assessing pharmaceutical adherence, one should adjust overlapping dates forward for a specified group (e.g. patient ids or medication classes) so that there is no overlap in days supply. For example, if a patient receives a 30 days supply on January 1st, and another 15 days later, the next fill date should be moved up 15 days. This function is modeled after recommendations from Canfield SL, Zuckerman A, Anguiano RH, Jolly JA, DeClercq J. Navigating the wild west of medication adherence reporting in specialty pharmacy. J Manag Care Spec Pharm. 2019;25(10):1073-77.

Usage

```
propagate_date(.data, .date_var = NULL, .days_supply_var = NULL)
```

Arguments

`.data`	Data to be piped into the function
`.date_var`	Date, column indicating the date of a given fill
`.days_supply_var`	Integer, column indicating the days supply of a given fill

Value

The initial claims data frame with an appended column, `adjusted_date`

Note

This function relies on anydate to parse the users date variable into a date class. So, for most columns passed to .date_var, this function will run without warning or error. For example, `anydate(30)` will return "1970-01-31" even though 30 is most likely a days supply. If strange results are produced, double check that the date variable being specified is indeed a fill date.

Examples

```
library(adheRenceRX)
library(dplyr)

toy_claims %>%
  filter(ID == "D") %>%
  propagate_date(.date_var = date, .days_supply_var = days_supply)
```

rank_episodes *Rank Episodes of Care*

Description

This function identifies and labels all episodes of care for a given patient in chronological order. A new episode begins after a specified gap in therapy has occurred. It is meant to be used after one has appropriately adjusted dates (`propagate_date()`) and identified gaps in therapy (`identify_gaps()`).

Usage

```
rank_episodes(.data, .permissible_gap = NULL, .initial_rank = 1)
```

Arguments

`.data`	Data frame with a "gap" column appended from `identify_gaps()`
`.permissible_gap`	Integer value suggesting the maximum gap allowed before labeling a new episode of care
`.initial_rank`	Integer value to identify what the indexing rank should be (defaults to 1).

Value

The initial claims data frame with an `episode` column appended, which ranks episodes of care in time

Note

This function assumes an `adjusted_date` column, which is produced by the `propagate_date()` function and a `gap` column, which is produced by `identify_gaps()`. If you would like to rank episodes of care using other dates and a separate column for gaps, you'll need to rename those columns before passing the frame to `rank_episodes()`. Notably, this is on purpose as this step should almost always come after the former two.

Examples

```
library(adheRenceRX)
library(dplyr)

toy_claims %>%
  filter(ID == "D") %>%
  propagate_date() %>%
  identify_gaps() %>%
  rank_episodes(.permissible_gap = 20, .initial_rank = 1)
```

summarise_gaps *Summarise Gaps in Therapy*

Description

This function serves as a convenience wrapper of `dplyr::summarise()`, which takes the grouped variables and summarises their gaps in therapy. This function is to be used after `propagate_date()`.

Usage

```
summarise_gaps(.data)
```

Arguments

`.data`	Data to be piped into the function

Value

A summary of gaps in therapy

Note

This function relies an `adjusted_date` column to identify gaps in therapy. So, if you don't want to use `propagate_date()` beforehand, you'll need to rename the date variable you wish to use to `adjusted_date`.

Examples

```
library(adheRenceRX)
library(dplyr)

toy_claims %>%
  filter(ID == "D") %>%
  propagate_date(.date_var = date, .days_supply_var = days_supply) %>%
  summarise_gaps()
```

toy_claims *A Toy Patient Medication Claims* tibble

Description

This mock patient claims tibble is meant to test adheRenceRX with scenarios presented in Figure 1. of Canfield SL, Zuckerman A, Anguiano RH, Jolly JA, DeClercq J. Navigating the wild west of medication adherence reporting in specialty pharmacy. J Manag Care Spec Pharm. 2019;25(10):1073-77

Usage

```
toy_claims
```

Format

A tibble with 22 rows and 3 variables:

ID patient ID to be grouped by

date date of claim

days_supply number of days supplied

Index