Data_cleaning_e3

Teodor Chakarov

2022-03-28

Contents

Tutorium in R

Data Cleaning with R

Exercise Data Cleaning with R - Number 3

By: Teodor Chakarov 12141198

Converting data types and adding new column

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(assertive)
library(stringr)
library(ggplot2)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
```

```
##
##     date, intersect, setdiff, union
```

```
library(visdat)
```

Checking datatypes of our data set

```
bike_share_rides <- readRDS(file = "bike_share_rides_ch1_1.rds")

glimpse(bike_share_rides)
```

```
## Rows: 35,229
## Columns: 10
## $ ride_id         <int> 52797, 54540, 87695, 45619, 70832, 96135, 29928, 83331~
## $ date            <chr> "2017-04-15", "2017-04-19", "2017-04-14", "2017-04-03"~
## $ duration        <chr> "1316.15 minutes", "8.13 minutes", "24.85 minutes", "6~
## $ station_A_id    <dbl> 67, 21, 16, 58, 16, 6, 5, 16, 5, 81, 30, 16, 16, 67, 2~
## $ station_A_name  <chr> "San Francisco Caltrain Station 2  (Townsend St at 4th~
## $ station_B_id    <dbl> 89, 64, 355, 368, 81, 66, 350, 91, 62, 81, 109, 10, 80~
## $ station_B_name  <chr> "Division St at Potrero Ave", "5th St at Brannan St", ~
## $ bike_id         <dbl> 1974, 860, 2263, 1417, 507, 75, 388, 239, 1449, 3289, ~
## $ user_gender     <chr> "Male", "Male", "Male", "Male", "Male", "Male", "Male"~
## $ user_birth_year <dbl> 1972, 1986, 1993, 1981, 1981, 1988, 1993, 1996, 1993, ~
```

The summary of one column

```
summary(bike_share_rides$user_birth_year)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1900    1979    1986    1984    1991    2001
```

```
# Convert user_birth_year to factor: user_birth_year_fct
bike_share_rides <- bike_share_rides %>%
  mutate(user_birth_year_fct = as.factor(user_birth_year))

# Assert user_birth_year_fct is a factor
assert_is_factor(bike_share_rides$user_birth_year_fct)

# Summary of user_birth_year_fct
summary(bike_share_rides$user_birth_year_fct)
```

```
## 1900 1902 1923 1931 1938 1939 1941 1942 1943 1945 1946 1947 1948 1949 1950 1951
##    1    7    2   23    2    1    3   10    4   16    5   24    9   30   37   25
## 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967
##   70   49   65   66  112   62  156   99  196  161  256  237  245  349  225  363
## 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983
##  365  331  370  548  529  527  563  601  481  541  775  876  825 1016 1056 1262
## 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999
## 1157 1318 1606 1672 2135 1872 2062 1582 1703 1498 1476 1185  813  358  365  348
## 2000 2001
##  473   30
```

```
glimpse(bike_share_rides)
```

```
## Rows: 35,229
## Columns: 11
## $ ride_id             <int> 52797, 54540, 87695, 45619, 70832, 96135, 29928, 8~
## $ date                <chr> "2017-04-15", "2017-04-19", "2017-04-14", "2017-04~
```

```
## $ duration            <chr> "1316.15 minutes", "8.13 minutes", "24.85 minutes"~
## $ station_A_id        <dbl> 67, 21, 16, 58, 16, 6, 5, 16, 5, 81, 30, 16, 16, 6~
## $ station_A_name      <chr> "San Francisco Caltrain Station 2  (Townsend St at~
## $ station_B_id        <dbl> 89, 64, 355, 368, 81, 66, 350, 91, 62, 81, 109, 10~
## $ station_B_name      <chr> "Division St at Potrero Ave", "5th St at Brannan S~
## $ bike_id             <dbl> 1974, 860, 2263, 1417, 507, 75, 388, 239, 1449, 32~
## $ user_gender         <chr> "Male", "Male", "Male", "Male", "Male", "Male", "M~
## $ user_birth_year     <dbl> 1972, 1986, 1993, 1981, 1981, 1988, 1993, 1996, 19~
## $ user_birth_year_fct <fct> 1972, 1986, 1993, 1981, 1981, 1988, 1993, 1996, 19~
```

We can see that we have new column which is factorial of the previous double column.

Trimming strings

```
bike_share_rides <- bike_share_rides %>%
  # Remove 'minutes' from duration: duration_trimmed
  mutate(duration_trimmed = str_remove(duration, " minutes"), #add new col with removed str
         # Convert duration_trimmed to numeric: duration_mins
         duration_mins = as.numeric(duration_trimmed))

# Glimpse at bike_share_rides
glimpse(bike_share_rides)
```

```
## Rows: 35,229
## Columns: 13
## $ ride_id             <int> 52797, 54540, 87695, 45619, 70832, 96135, 29928, 8~
## $ date                <chr> "2017-04-15", "2017-04-19", "2017-04-14", "2017-04~
## $ duration            <chr> "1316.15 minutes", "8.13 minutes", "24.85 minutes"~
## $ station_A_id        <dbl> 67, 21, 16, 58, 16, 6, 5, 16, 5, 81, 30, 16, 16, 6~
## $ station_A_name      <chr> "San Francisco Caltrain Station 2  (Townsend St at~
## $ station_B_id        <dbl> 89, 64, 355, 368, 81, 66, 350, 91, 62, 81, 109, 10~
## $ station_B_name      <chr> "Division St at Potrero Ave", "5th St at Brannan S~
## $ bike_id             <dbl> 1974, 860, 2263, 1417, 507, 75, 388, 239, 1449, 32~
## $ user_gender         <chr> "Male", "Male", "Male", "Male", "Male", "Male", "M~
## $ user_birth_year     <dbl> 1972, 1986, 1993, 1981, 1981, 1988, 1993, 1996, 19~
## $ user_birth_year_fct <fct> 1972, 1986, 1993, 1981, 1981, 1988, 1993, 1996, 19~
## $ duration_trimmed    <chr> "1316.15", "8.13", "24.85", "6.35", "9.8", "17.47"~
## $ duration_mins       <dbl> 1316.15, 8.13, 24.85, 6.35, 9.80, 17.47, 16.52, 14~
```

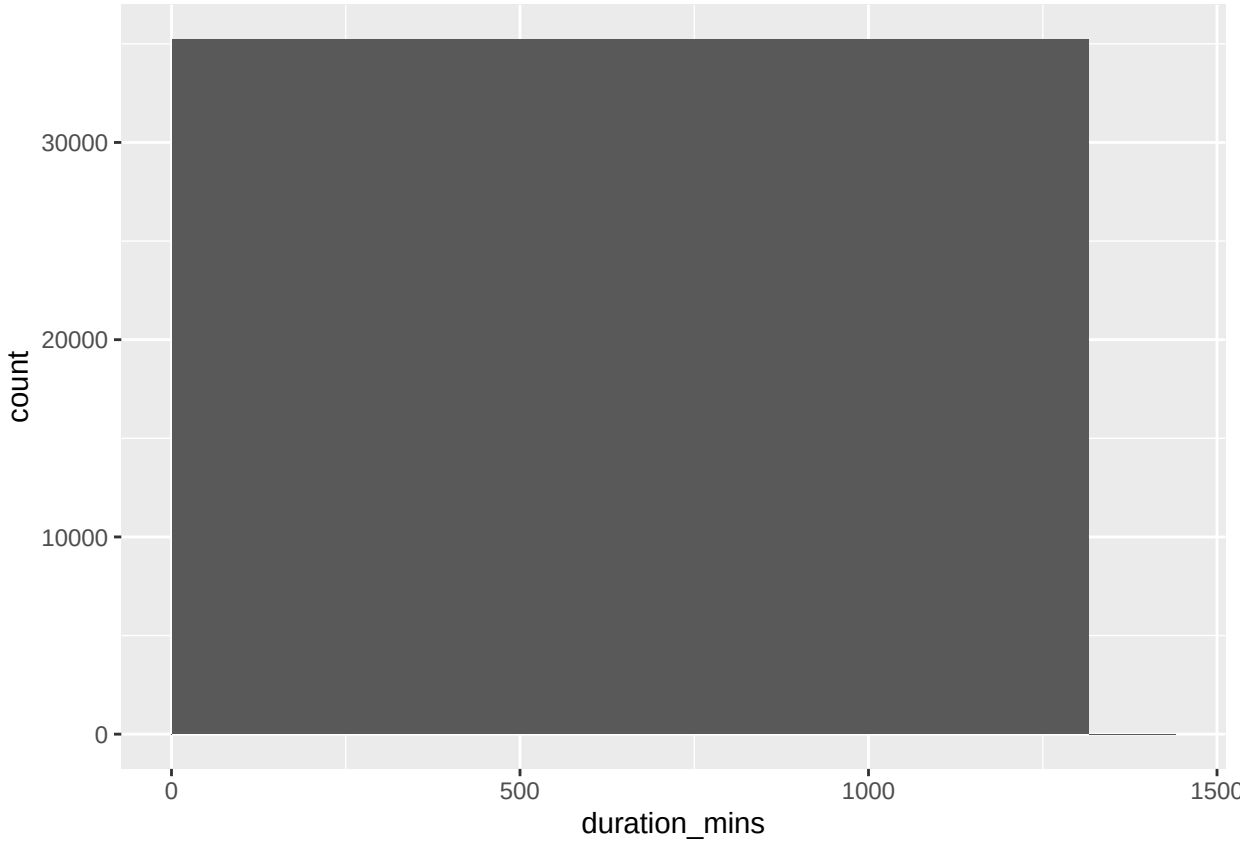
```
# Assert duration_mins is numeric
assert_is_numeric(bike_share_rides$duration_mins)

# Calculate mean duration
mean(bike_share_rides$duration_mins)
```

```
## [1] 13.06214
```

```
# Create breaks
breaks <- c(min(bike_share_rides$duration_mins), 0, 1440, max(bike_share_rides$duration_mins))

# Create a histogram of duration_min
ggplot(bike_share_rides, aes(duration_mins)) +
  geom_histogram(breaks = breaks)
```



Working selecting data, duplicats

Adding new column in which values above 1440 will be replaced with max 1440:

```
# duration_min_const: replace vals of duration_min > 1440 with 1440
bike_share_rides <- bike_share_rides %>%
  mutate(duration_min_const = replace(duration_mins, duration_mins>1440, 1440))

# Make sure all values of duration_min_const are between 0 and 1440
assert_all_are_in_closed_range(bike_share_rides$duration_min_const, lower = 0, upper = 1440)
```

Convert datetime column and select the needed data

```
# Convert date to Date type
bike_share_rides <- bike_share_rides %>%
  mutate(date = as.Date(date))

# Filter for rides that occurred before or on today's date
bike_share_rides_past <- bike_share_rides %>%
  filter(date <= today())
```

Remove duplicated data

```
#Count the number of full duplicates
sum(duplicated(bike_share_rides))
```

```
## [1] 0
```

```
#Remove duplicates
bike_share_rides_unique <- distinct(bike_share_rides)

#Count the full duplicates in bike_share_rides_unique
sum(duplicated(bike_share_rides_unique))
```

```
## [1] 0
```

```
# Find duplicated ride_ids
bike_share_rides %>%
  # Count the number of occurrences of each ride_id
  count(ride_id) %>%
  # Filter for rows with a count > 1
  filter(n > 1)
```

```
## # A tibble: 0 x 2
## # ... with 2 variables: ride_id <int>, n <int>
```

```
# Remove full and partial duplicates
bike_share_rides_unique <- bike_share_rides %>%
  # Only based on ride_id instead of all cols
  distinct(ride_id, .keep_all = TRUE)
```

```
bike_share_rides_unique %>%
  # Count the number of occurrences of each ride_id
  count(ride_id) %>%
  # Filter for rows with a count > 1
  filter(n > 1)
```

```
## # A tibble: 0 x 2
## # ... with 2 variables: ride_id <int>, n <int>
```

Grouping by:

```
bike_share_rides %>%
  # Group by ride_id and date
  group_by(ride_id, date) %>%
  # Add duration_min_avg column
  mutate(duration_min_avg = mean(duration_mins)) %>%
  # Remove duplicates based on ride_id and date, keep all cols
  distinct(ride_id, date, .keep_all = TRUE) %>%
  # Remove duration_min column
  select(-duration_mins)
```

```
## # A tibble: 35,229 x 14
## # Groups:   ride_id, date [35,229]
##    ride_id date       duration        station_A_id station_A_name   station_B_id
##      <int> <date>     <chr>                  <dbl> <chr>                   <dbl>
##  1   52797 2017-04-15 1316.15 minutes           67 San Francisco C~           89
##  2   54540 2017-04-19 8.13 minutes              21 Montgomery St B~           64
##  3   87695 2017-04-14 24.85 minutes             16 Steuart St at M~          355
##  4   45619 2017-04-03 6.35 minutes              58 Market St at 10~          368
##  5   70832 2017-04-10 9.8 minutes               16 Steuart St at M~           81
##  6   96135 2017-04-18 17.47 minutes              6 The Embarcadero~           66
##  7   29928 2017-04-22 16.52 minutes              5 Powell St BART ~          350
##  8   83331 2017-04-11 14.72 minutes             16 Steuart St at M~           91
##  9   72424 2017-04-05 4.12 minutes               5 Powell St BART ~           62
```

```
## 10   25910 2017-04-20 25.77 minutes             81 Berry St at 4th~           81
## # ... with 35,219 more rows, and 8 more variables: station_B_name <chr>,
## #   bike_id <dbl>, user_gender <chr>, user_birth_year <dbl>,
## #   user_birth_year_fct <fct>, duration_trimmed <chr>,
## #   duration_min_const <dbl>, duration_min_avg <dbl>
```

```
sfo_survey <- readRDS(file = "sfo_survey_ch2_1.rds")
```

```
# Count the number of occurrences of dest_size
sfo_survey %>%
  count(dest_size)
```

```
##   dest_size    n
## 1   Small      1
## 2       Hub    1
## 3       Hub 1756
## 4     Large  143
## 5   Large      1
## 6    Medium  682
## 7     Small  225
```

```
# Count the number of occurrences of dest_size
sfo_survey %>%
  count(dest_size)
```

```
##   dest_size    n
## 1   Small      1
## 2       Hub    1
## 3       Hub 1756
## 4     Large  143
## 5   Large      1
## 6    Medium  682
## 7     Small  225
```

String data manipulation

Adding new column with trimmed whitespace and lowercase chars

```
# Add new columns to sfo_survey
sfo_survey <- sfo_survey %>%
  # dest_size_trimmed: dest_size without whitespace
  mutate(dest_size_trimmed = str_trim(dest_size),
         # cleanliness_lower: cleanliness converted to lowercase
         cleanliness_lower = str_to_lower(cleanliness))

# Count values of dest_size_trimmed
sfo_survey %>%
  count(dest_size_trimmed)
```

```
##   dest_size_trimmed    n
## 1               Hub 1757
## 2             Large  144
## 3            Medium  682
## 4             Small  226
```

```
# Count values of cleanliness_lower
sfo_survey %>%
```

```
  count(cleanliness_lower)
```

```
##   cleanliness_lower    n
## 1           average  433
## 2             clean  970
## 3             dirty    2
## 4    somewhat clean 1254
## 5    somewhat dirty   30
## 6              <NA>  120
```

When we have patterns, regex symbols, to escape them we use fixed():

```
# Filter for rows with "(" or ")" in the phone column
#sfo_survey %>%
#  filter(str_detect(phone, fixed("(")) | str_detect(phone, fixed(")")))
```

Remove the unnecessary chars and add new column in which we get rid of "-"

```
# Remove parentheses from phone column
#phone_no_parens <- sfo_survey$phone %>%
#  # Remove "("s
#  str_remove_all(fixed("(")) %>%
#  # Remove ")"s
#  str_remove_all(fixed(")"))

# Add phone_no_parens as column
#sfo_survey %>%
#  mutate(phone_no_parens = phone_no_parens,
#  # Replace all hyphens in phone_no_parens with spaces
#         phone_clean = str_replace_all(phone_no_parens,"-", " "))
```

Check if sting is = 12 and removing the invalid strings

```
# Check out the invalid numbers
#sfo_survey %>%
#  filter(str_length(phone) != 12)

# Remove rows with invalid numbers
#sfo_survey %>%
#  filter(str_length(phone) == 12)
```

Data uniformity

To change the format of the date column

```
accounts <- readRDS(file = "ch3_1_accounts.rds")
account_offices <- read.csv(file = "account_offices.csv")

formats <- c("%Y-%m-%d", "%B %d, %Y")

# Convert dates to the same format
accounts %>%
  mutate(date_opened_clean = parse_date_time(date_opened, formats))
```

```
##          id        date_opened    total date_opened_clean
## 1  A880C79F         2003-10-19   169305        2003-10-19
## 2  BE8222DF   October 05, 2018   107460        2018-10-05
```

```
## 3  19F9E113         2008-07-29 15297152        2008-07-29
## 4  A2FE52A3         2005-06-09 14897272        2005-06-09
## 5  F6DC2C08         2012-03-31   124568        2012-03-31
## 6  D2E55799         2007-06-20 13635752        2007-06-20
## 7  53AE87EF  December 01, 2017 15375984        2017-12-01
## 8  3E97F253         2019-06-03 14515800        2019-06-03
## 9  4AE79EA1         2011-05-07 23338536        2011-05-07
## 10 2322DFB4         2018-04-07   189524        2018-04-07
## 11 645335B2         2018-11-16   154001        2018-11-16
## 12 D5EB0F00         2001-04-16   174576        2001-04-16
## 13 1EB593F7         2005-04-21   191989        2005-04-21
## 14 DDBA03D9         2006-06-13  9617192        2006-06-13
## 15 40E4A2F4         2009-01-07   180547        2009-01-07
## 16 39132EEA         2012-07-07 15611960        2012-07-07
## 17 387F8E4D   January 03, 2011  9402640        2011-01-03
## 18 11C3C3C0  December 24, 2017   180003        2017-12-24
## 19 C2FC91E1         2004-05-21   105722        2004-05-21
## 20 FB8F01C1         2001-09-06 22575072        2001-09-06
## 21 0128D2D0         2005-04-09 19179784        2005-04-09
## 22 BE6E4B3F         2009-10-20 15679976        2009-10-20
## 23 7C6E2ECC         2003-05-16   169814        2003-05-16
## 24 02E63545         2015-10-25   125117        2015-10-25
## 25 4399C98B       May 19, 2001   130421        2001-05-19
## 26 98F4CF0F       May 27, 2014 14893944        2014-05-27
## 27 247222A6       May 26, 2015   150372        2015-05-26
## 28 420985EE         2008-12-27   123125        2008-12-27
## 29 0E3903BA         2015-11-11   182668        2015-11-11
## 30 64EF994F         2009-02-26   161141        2009-02-26
## 31 CCF84EDB         2008-12-26   136128        2008-12-26
## 32 51C21705     April 22, 2016 16191136        2016-04-22
## 33 C868C6AD   January 31, 2000 11733072        2000-01-31
## 34 92C237C6         2005-12-13 11838528        2005-12-13
## 35 9ECEADB2       May 17, 2018   146153        2018-05-17
## 36 DF0AFE50         2004-12-03 15250040        2004-12-03
## 37 5CD605B3         2016-10-19    87921        2016-10-19
## 38 402839E2 September 14, 2019   163416        2019-09-14
## 39 78286CE7         2009-10-05 15049216        2009-10-05
## 40 168E071B         2013-07-11    87826        2013-07-11
## 41 466CCDAA         2002-03-24 14981304        2002-03-24
## 42 8DE1ECB9         2015-10-17   217975        2015-10-17
## 43 E19FE6B5      June 06, 2009   101936        2009-06-06
## 44 1240D39C September 07, 2011 15761824        2011-09-07
## 45 A7BFAA72         2019-11-12   133790        2019-11-12
## 46 C3D24436       May 24, 2002   101584        2002-05-24
## 47 FAD92F0F September 13, 2007 17081064        2007-09-13
## 48 236A1D51         2019-10-01 18486936        2019-10-01
## 49 A6DDDC4C         2000-08-17    67962        2000-08-17
## 50 DDFD0B3D         2001-04-11 15776384        2001-04-11
## 51 D13375E9  November 01, 2005 13944632        2005-11-01
## 52 AC50B796         2016-06-30 16111264        2016-06-30
## 53 290319FD       May 27, 2005   170178        2005-05-27
## 54 FC71925A  November 02, 2006   186281        2006-11-02
## 55 7B0F3685         2013-05-23   179102        2013-05-23
## 56 BE411172         2017-02-24 17689984        2017-02-24
```

```
## 57 58066E39 September 16, 2015 17025632        2015-09-16
## 58 EA7FF83A         2004-11-02 11598704        2004-11-02
## 59 14A2DDB7         2019-03-06 12808952        2019-03-06
## 60 305EEAA8         2018-09-01 14417728        2018-09-01
## 61 8F25E54C  November 24, 2008   189126        2008-11-24
## 62 19DD73C6         2002-12-31 14692600        2002-12-31
## 63 ACB8E6AF         2013-07-27    71359        2013-07-27
## 64 91BFCC40         2014-01-10   132859        2014-01-10
## 65 86ACAF81         2011-12-14 24533704        2011-12-14
## 66 77E85C14  November 20, 2009 13868192        2009-11-20
## 67 C5C6B79D         2008-03-01   188424        2008-03-01
## 68 0E5B69F5         2018-05-07 18650632        2018-05-07
## 69 5275B518         2017-11-23    71665        2017-11-23
## 70 17217048       May 25, 2001 20111208        2001-05-25
## 71 E7496A7F         2008-09-27   142669        2008-09-27
## 72 41BBB7B4  February 22, 2005   144229        2005-02-22
## 73 F6C7ABA1         2008-01-07   183440        2008-01-07
## 74 E699DF01  February 17, 2008   199603        2008-02-17
## 75 BACA7378         2005-05-11   204271        2005-05-11
## 76 84A4302F         2003-08-12 19420648        2003-08-12
## 77 F8A78C27     April 05, 2006    41164        2006-04-05
## 78 8BADDF6A  December 31, 2010   158203        2010-12-31
## 79 9FB57E68 September 01, 2017   216352        2017-09-01
## 80 5C98E8F5         2014-11-25   103200        2014-11-25
## 81 6BB53C2A  December 03, 2016   146394        2016-12-03
## 82 E23F2505   October 15, 2017   121614        2017-10-15
## 83 0C121914      June 21, 2017   227729        2017-06-21
## 84 3627E08A         2008-04-01   238104        2008-04-01
## 85 A94493B3    August 01, 2009    85975        2009-08-01
## 86 0682E9DE         2002-10-01    72832        2002-10-01
## 87 49931170         2011-03-25 14519856        2011-03-25
## 88 A154F63B         2000-07-11   133800        2000-07-11
## 89 3690CCED         2014-10-19   226595        2014-10-19
## 90 48F5E6D8  February 16, 2020   135435        2020-02-16
## 91 515FAD84         2013-06-20    98190        2013-06-20
## 92 59794264         2008-01-16   157964        2008-01-16
## 93 2038185B         2016-06-24   194662        2016-06-24
## 94 65EAC615  February 20, 2004   140191        2004-02-20
## 95 6C7509C9 September 16, 2000   212089        2000-09-16
## 96 BD969A9D         2007-04-29   167238        2007-04-29
## 97 B0CDCE3D       May 28, 2014   145240        2014-05-28
## 98 33A7F03E   October 14, 2007   191839        2007-10-14
```

```
head(account_offices)
```

```
##         id   office
## 1 A880C79F New York
## 2 BE8222DF New York
## 3 19F9E113    Tokyo
## 4 A2FE52A3    Tokyo
## 5 F6DC2C08 New York
## 6 D2E55799    Tokyo
```

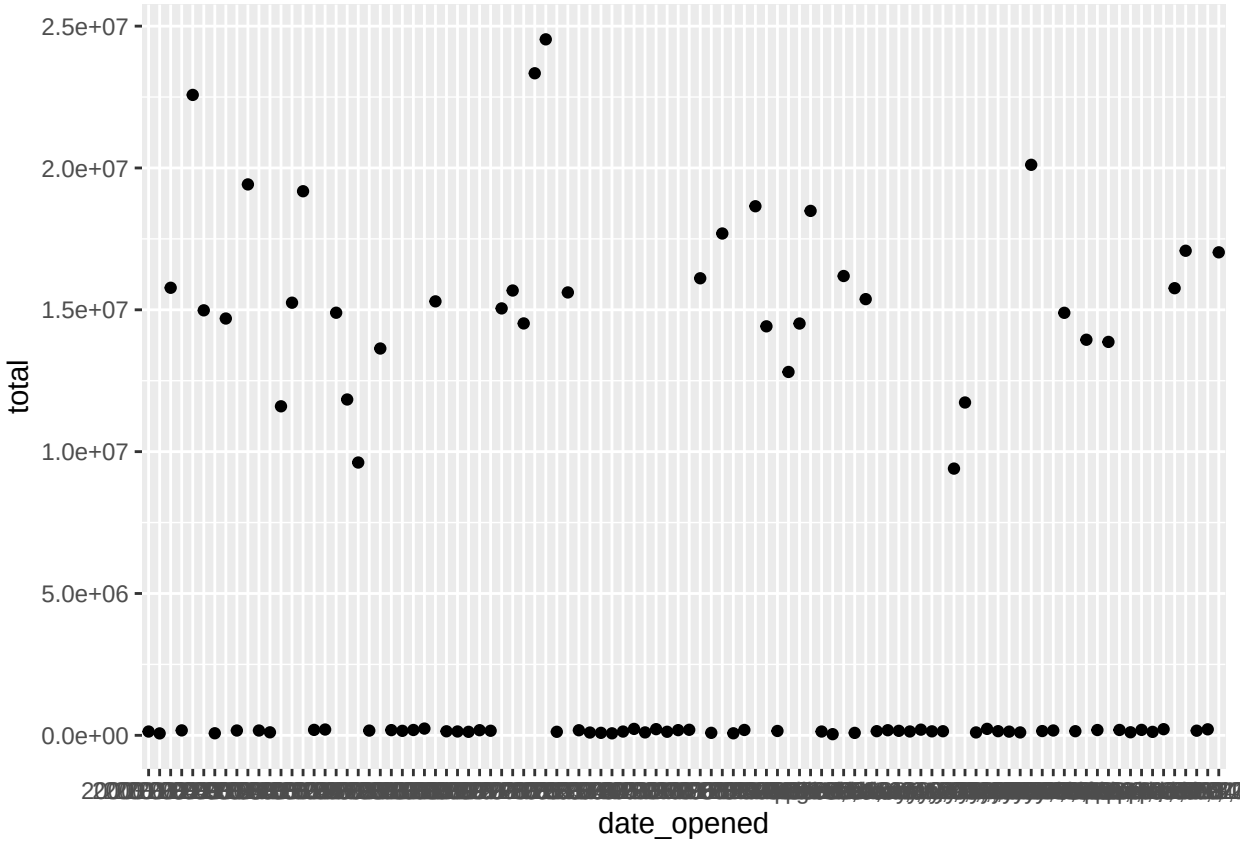
```
accounts %>%
  left_join(account_offices, by = "id")
```

```
##          id        date_opened    total   office
## 1  A880C79F         2003-10-19   169305 New York
## 2  BE8222DF   October 05, 2018   107460 New York
## 3  19F9E113         2008-07-29 15297152    Tokyo
## 4  A2FE52A3         2005-06-09 14897272    Tokyo
## 5  F6DC2C08         2012-03-31   124568 New York
## 6  D2E55799         2007-06-20 13635752    Tokyo
## 7  53AE87EF  December 01, 2017 15375984     <NA>
## 8  3E97F253         2019-06-03 14515800     <NA>
## 9  4AE79EA1         2011-05-07 23338536     <NA>
## 10 2322DFB4         2018-04-07   189524     <NA>
## 11 645335B2         2018-11-16   154001     <NA>
## 12 D5EB0F00         2001-04-16   174576     <NA>
## 13 1EB593F7         2005-04-21   191989     <NA>
## 14 DDBA03D9         2006-06-13  9617192     <NA>
## 15 40E4A2F4         2009-01-07   180547     <NA>
## 16 39132EEA         2012-07-07 15611960     <NA>
## 17 387F8E4D   January 03, 2011  9402640     <NA>
## 18 11C3C3C0  December 24, 2017   180003     <NA>
## 19 C2FC91E1         2004-05-21   105722     <NA>
## 20 FB8F01C1         2001-09-06 22575072     <NA>
## 21 0128D2D0         2005-04-09 19179784     <NA>
## 22 BE6E4B3F         2009-10-20 15679976     <NA>
## 23 7C6E2ECC         2003-05-16   169814     <NA>
## 24 02E63545         2015-10-25   125117     <NA>
## 25 4399C98B       May 19, 2001   130421     <NA>
## 26 98F4CF0F       May 27, 2014 14893944     <NA>
## 27 247222A6       May 26, 2015   150372     <NA>
## 28 420985EE         2008-12-27   123125     <NA>
## 29 0E3903BA         2015-11-11   182668     <NA>
## 30 64EF994F         2009-02-26   161141     <NA>
## 31 CCF84EDB         2008-12-26   136128     <NA>
## 32 51C21705     April 22, 2016 16191136     <NA>
## 33 C868C6AD   January 31, 2000 11733072     <NA>
## 34 92C237C6         2005-12-13 11838528     <NA>
## 35 9ECEADB2       May 17, 2018   146153     <NA>
## 36 DF0AFE50         2004-12-03 15250040     <NA>
## 37 5CD605B3         2016-10-19    87921     <NA>
## 38 402839E2 September 14, 2019   163416     <NA>
## 39 78286CE7         2009-10-05 15049216     <NA>
## 40 168E071B         2013-07-11    87826     <NA>
## 41 466CCDAA         2002-03-24 14981304     <NA>
## 42 8DE1ECB9         2015-10-17   217975     <NA>
## 43 E19FE6B5      June 06, 2009   101936     <NA>
## 44 1240D39C September 07, 2011 15761824     <NA>
## 45 A7BFAA72         2019-11-12   133790     <NA>
## 46 C3D24436       May 24, 2002   101584     <NA>
## 47 FAD92F0F September 13, 2007 17081064     <NA>
## 48 236A1D51         2019-10-01 18486936     <NA>
## 49 A6DDDC4C         2000-08-17    67962     <NA>
## 50 DDFD0B3D         2001-04-11 15776384     <NA>
## 51 D13375E9  November 01, 2005 13944632     <NA>
## 52 AC50B796         2016-06-30 16111264     <NA>
## 53 290319FD       May 27, 2005   170178     <NA>
```

```
## 54 FC71925A  November 02, 2006   186281     <NA>
## 55 7B0F3685         2013-05-23   179102     <NA>
## 56 BE411172         2017-02-24 17689984     <NA>
## 57 58066E39 September 16, 2015 17025632     <NA>
## 58 EA7FF83A         2004-11-02 11598704     <NA>
## 59 14A2DDB7         2019-03-06 12808952     <NA>
## 60 305EEAA8         2018-09-01 14417728     <NA>
## 61 8F25E54C  November 24, 2008   189126     <NA>
## 62 19DD73C6         2002-12-31 14692600     <NA>
## 63 ACB8E6AF         2013-07-27    71359     <NA>
## 64 91BFCC40         2014-01-10   132859     <NA>
## 65 86ACAF81         2011-12-14 24533704     <NA>
## 66 77E85C14  November 20, 2009 13868192     <NA>
## 67 C5C6B79D         2008-03-01   188424     <NA>
## 68 0E5B69F5         2018-05-07 18650632     <NA>
## 69 5275B518         2017-11-23    71665     <NA>
## 70 17217048       May 25, 2001 20111208     <NA>
## 71 E7496A7F         2008-09-27   142669     <NA>
## 72 41BBB7B4  February 22, 2005   144229     <NA>
## 73 F6C7ABA1         2008-01-07   183440     <NA>
## 74 E699DF01  February 17, 2008   199603     <NA>
## 75 BACA7378         2005-05-11   204271     <NA>
## 76 84A4302F         2003-08-12 19420648     <NA>
## 77 F8A78C27     April 05, 2006    41164     <NA>
## 78 8BADDF6A  December 31, 2010   158203     <NA>
## 79 9FB57E68 September 01, 2017   216352     <NA>
## 80 5C98E8F5         2014-11-25   103200     <NA>
## 81 6BB53C2A  December 03, 2016   146394     <NA>
## 82 E23F2505   October 15, 2017   121614     <NA>
## 83 0C121914      June 21, 2017   227729     <NA>
## 84 3627E08A         2008-04-01   238104     <NA>
## 85 A94493B3    August 01, 2009    85975     <NA>
## 86 0682E9DE         2002-10-01    72832     <NA>
## 87 49931170         2011-03-25 14519856     <NA>
## 88 A154F63B         2000-07-11   133800     <NA>
## 89 3690CCED         2014-10-19   226595     <NA>
## 90 48F5E6D8  February 16, 2020   135435     <NA>
## 91 515FAD84         2013-06-20    98190     <NA>
## 92 59794264         2008-01-16   157964     <NA>
## 93 2038185B         2016-06-24   194662     <NA>
## 94 65EAC615  February 20, 2004   140191     <NA>
## 95 6C7509C9 September 16, 2000   212089     <NA>
## 96 BD969A9D         2007-04-29   167238     <NA>
## 97 B0CDCE3D       May 28, 2014   145240     <NA>
## 98 33A7F03E   October 14, 2007   191839     <NA>
```

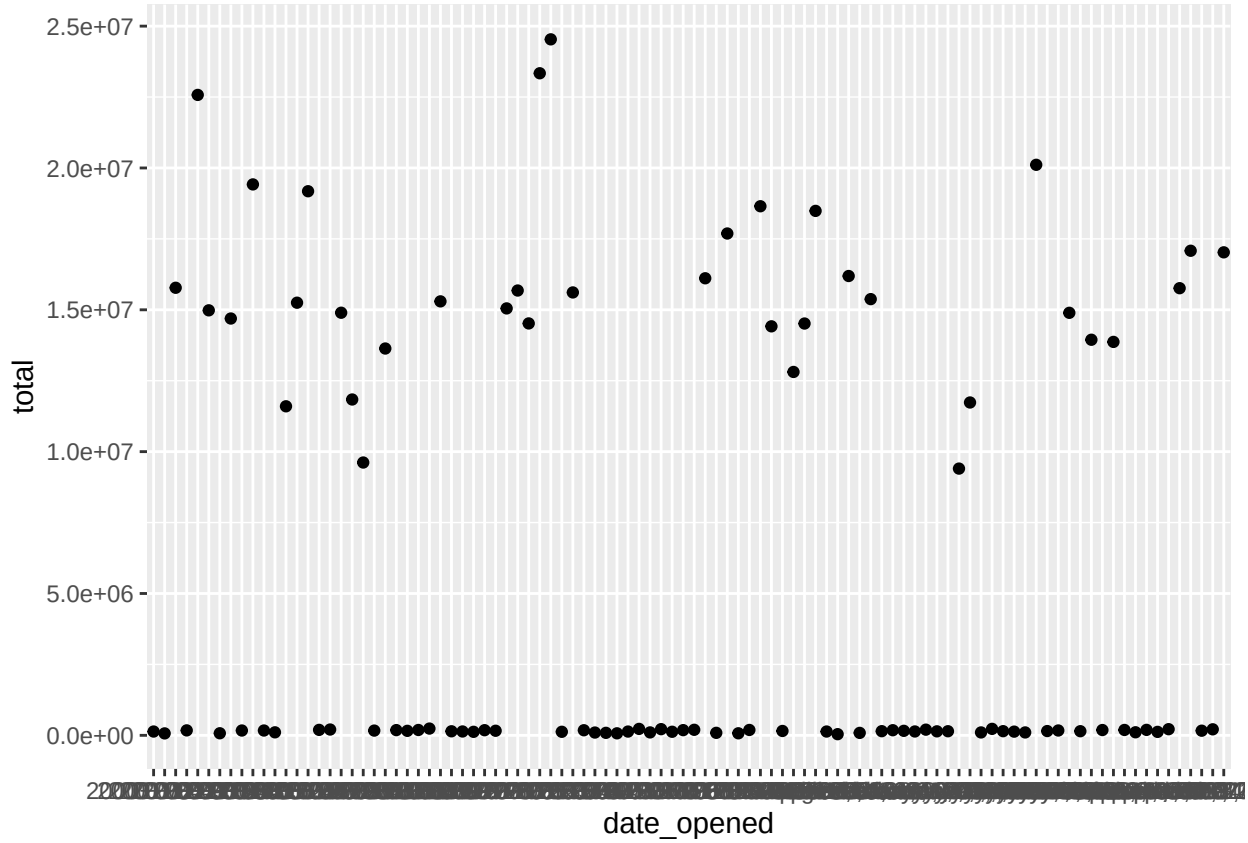
Visualizing data:

```
# Scatter plot of opening date and total amount
accounts %>%
  ggplot(aes(x = date_opened, y = total)) +
  geom_point()
```



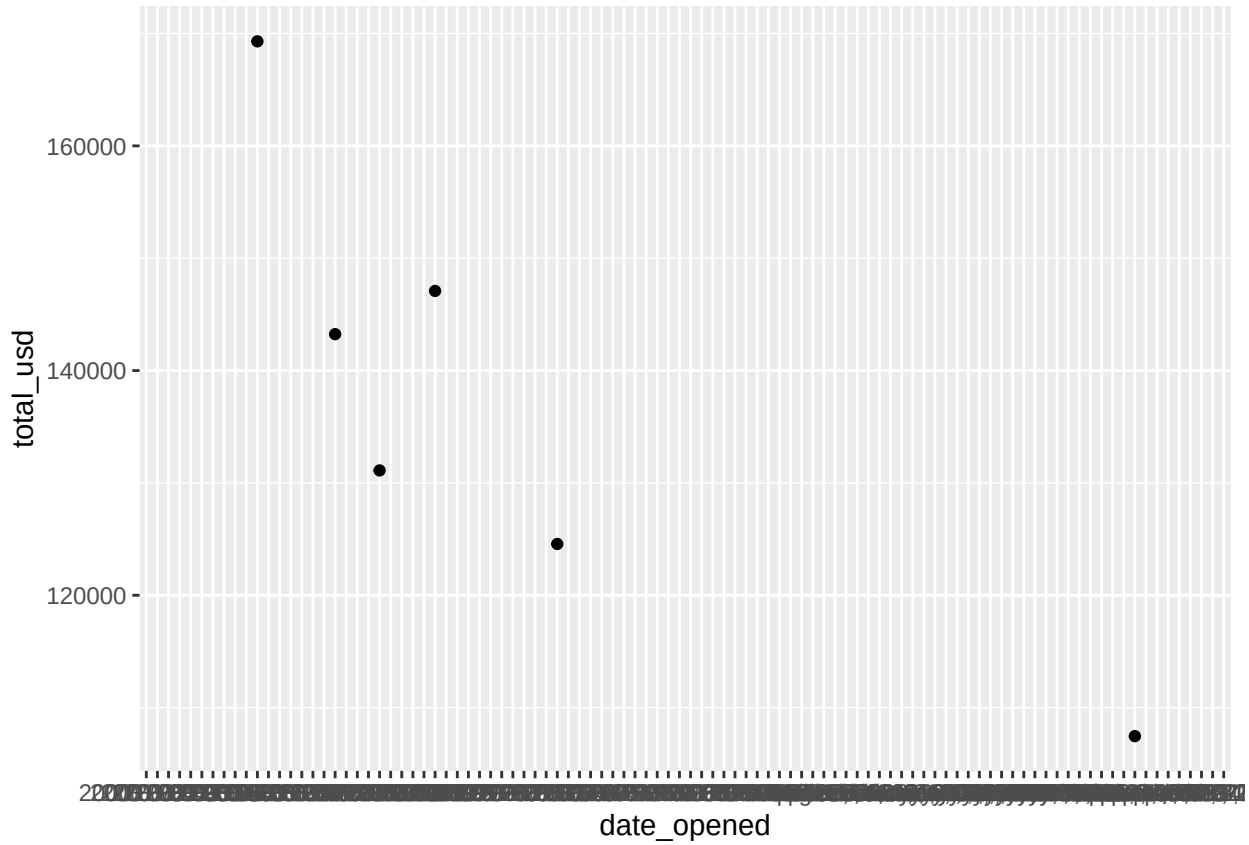
Visualizing the data, after thet im joining the tables by id and converting if office in Tokyo -> convert the currency to USD

```
# Scatter plot of opening date and total amount
accounts %>%
  ggplot(aes(x = date_opened, y = total)) +
  geom_point()
```




```
# Left join accounts to account_offices by id
accounts %>%
  left_join(account_offices, by = "id") %>%
  # Convert totals from the Tokyo office to USD
  mutate(total_usd = ifelse(office == "Tokyo", total / 104, total)) %>%
  # Scatter plot of opening date vs total_usd
  ggplot(aes(x = date_opened, y = total_usd)) +
    geom_point()
```

```
## Warning: Removed 92 rows containing missing values (geom_point).
```



Filter and validate data.

```
# Find invalid totals
#accounts %>%
#  # theoretical_total: sum of the three funds
#  mutate(theoretical_total = fund_A + fund_B + fund_C) %>%
  # Find accounts where total doesn't match theoretical_total
#  filter(theoretical_total != total)
```

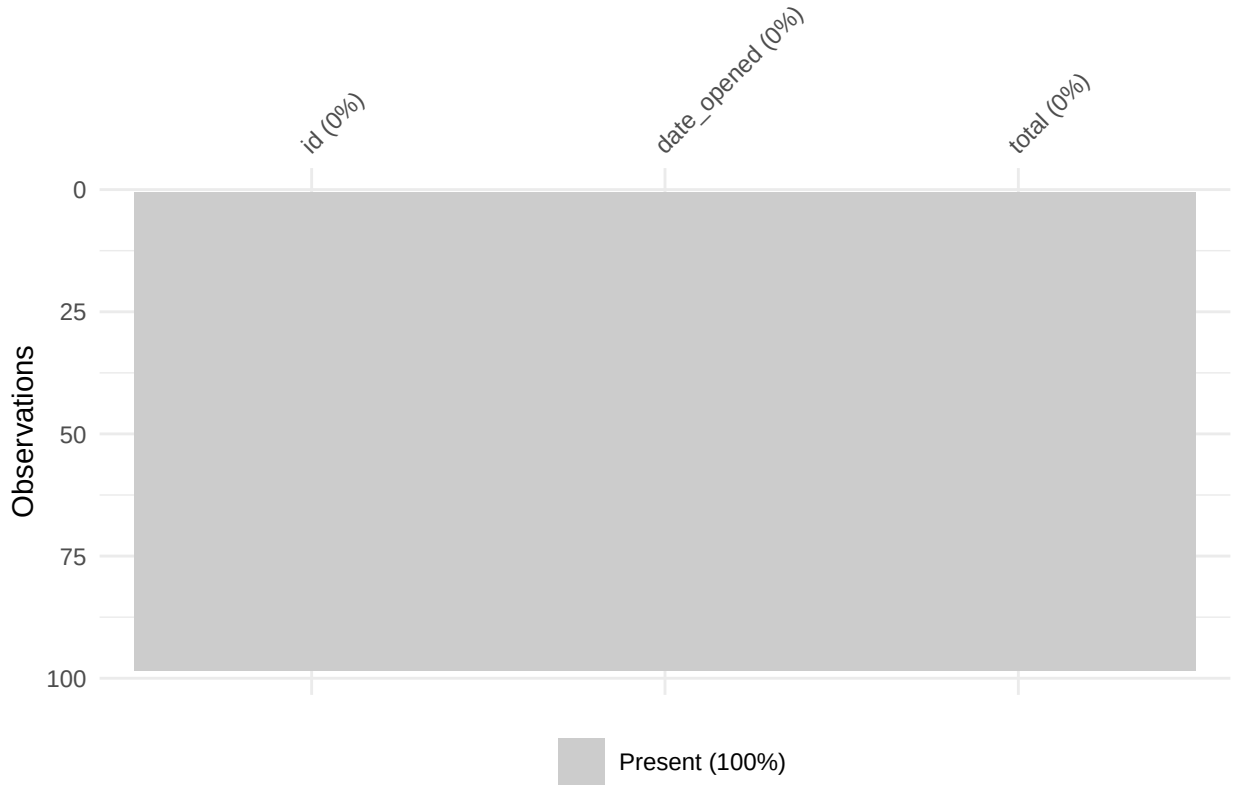
Turn to numeric and then - todays date

```
# Find invalid acct_age
#accounts %>%
  # theoretical_age: age of acct based on date_opened
#  mutate(theoretical_age = floor(as.numeric(date_opened %--% today(), "years"))) %>%
  # Filter for rows where acct_age is different from theoretical_age
#  filter(theoretical_age != acct_age)
```

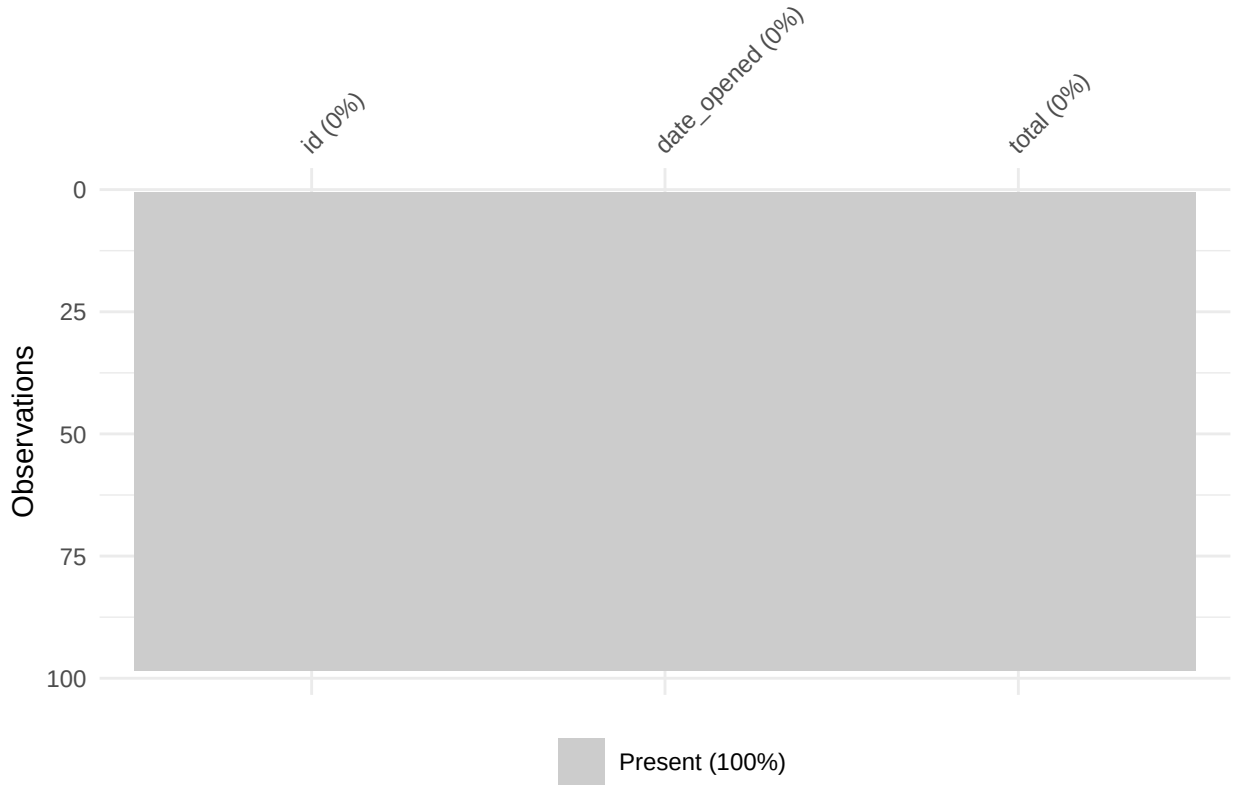
To visualize the missing data

```
vis_miss(accounts)
```

```
## Warning: `gather_()` was deprecated in tidyr 1.2.0.
## Please use `gather()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```




```
# Visualize the missing values by column
vis_miss(accounts)
```




```
#accounts %>%
  # missing_inv: Is inv_amount missing?
 # mutate(missing_inv = is.na(inv_amount)) %>%
  # Group by missing_inv
#  group_by(missing_inv) %>%
  # Calculate mean age for each missing_inv group
#  summarize(avg_age = mean(age))

# Sort by age and visualize missing vals
#accounts %>%
#  arrange(age) %>%
#  vis_miss()
```

Remove nan values

```
# Create accounts_clean
#accounts_clean <- accounts %>%
  # Filter to remove rows with missing cust_id
#  filter(!is.na(cust_id))

#accounts_clean
```

Create a new column called ***acct_amount_filled***, which contains the values of ***acct_amount***, except all NA values should be replaced with 5 times the amount in ***inv_amount***

```
# Create accounts_clean
#accounts_clean <- accounts %>%
  # Filter to remove rows with missing cust_id
```

```
#  filter(!is.na(cust_id)) %>%
  # Add new col acct_amount_filled with replaced NAs
#  mutate(acct_amount_filled = ifelse(is.na(acct_amount), inv_amount * 5, acct_amount))

#accounts_clean


# Assert that cust_id has no missing vals
#assert_all_are_not_na(accounts_clean$cust_id)
```

Comparing strings

With:

```
library("stringdist")
stringdist("string1", "sting2", method="lcs")
```

```
## [1] 3
```

Fixing typos:

```
library(fuzzyjoin)
zagat <- readRDS(file="zagat.rds")
fodors <- readRDS(file="fodors.rds")

# Count the number of each city variation
zagat %>%
  count(city)
```

```
##            city  n
## 1       atlanta 64
## 2   los angeles 72
## 3      new york 98
## 4     las vegas 26
## 5 san francisco 50
```

```
# Join and look at results
#zagat %>%
  # Left join based on stringdist using city and city_actual cols
#  stringdist_left_join(city, by = c("city" = "city_actual")) %>%
  # Select the name, city, and city_actual cols
#  select(name, city, city_actual)
```

Pair blocking

```
# Load reclin
library(reclin)
```

```
## Loading required package: lvec
```

```
##
## Attaching package: 'lvec'
```

```
## The following object is masked from 'package:base':
##
##     order
```

```
## Loading required package: ldat
```

```
## Loading required package: Rcpp
##
## Attaching package: 'ldat'
## The following objects are masked from 'package:base':
##
##     append, match, table, which
##
## Attaching package: 'reclin'
## The following object is masked from 'package:base':
##
##     identical
```

```
# Generate pairs with same city
pair_blocking(zagat, fodors, blocking_var = "city")
```

```
## Simple blocking
##   Blocking variable(s): city
##   First data set:  310 records
##   Second data set: 533 records
##   Total number of pairs: 40 532 pairs
##
## ldat with 40 532 rows and 2 columns
##         x   y
## 1       1   1
## 2       1   2
## 3       1   3
## 4       1   4
## 5       1   5
## 6       1   6
## 7       1   7
## 8       1   8
## 9       1   9
## 10      1  10
## :       :   :
## 40523 310 414
## 40524 310 415
## 40525 310 416
## 40526 310 417
## 40527 310 418
## 40528 310 419
## 40529 310 420
## 40530 310 421
## 40531 310 422
## 40532 310 423
```

```
# Generate pairs
pair_blocking(zagat, fodors, blocking_var = "city") %>%
  # Compare pairs by name using lcs()
  compare_pairs(by = "name",
                default_comparator = lcs())
```

```
## Compare
##   By: name
##
```

```
## Simple blocking
##   Blocking variable(s): city
##   First data set:  310 records
##   Second data set: 533 records
##   Total number of pairs: 40 532 pairs
##
## ldat with 40 532 rows and 3 columns
##         x   y      name
## 1       1   1 0.3157895
## 2       1   2 0.3225806
## 3       1   3 0.2307692
## 4       1   4 0.2608696
## 5       1   5 0.4545455
## 6       1   6 0.2142857
## 7       1   7 0.1052632
## 8       1   8 0.2222222
## 9       1   9 0.3000000
## 10      1  10 0.4516129
## :       :   :         :
## 40523 310 414 0.3606557
## 40524 310 415 0.2631579
## 40525 310 416 0.2105263
## 40526 310 417 0.3750000
## 40527 310 418 0.2978723
## 40528 310 419 0.2727273
## 40529 310 420 0.3437500
## 40530 310 421 0.3414634
## 40531 310 422 0.4081633
## 40532 310 423 0.1714286
```

```
# Generate pairs
pair_blocking(zagat, fodors, blocking_var = "city") %>%
  # Compare pairs by name, phone, addr
  compare_pairs(by = c("name", "phone", "addr"),
                default_comparator = jaro_winkler())
```

```
## Compare
##   By: name, phone, addr
##
## Simple blocking
##   Blocking variable(s): city
##   First data set:  310 records
##   Second data set: 533 records
##   Total number of pairs: 40 532 pairs
##
## ldat with 40 532 rows and 5 columns
##         x   y      name     phone      addr
## 1       1   1 0.4871062 0.6746032 0.5703661
## 2       1   2 0.5234025 0.5555556 0.6140351
## 3       1   3 0.4564103 0.7222222 0.5486355
## 4       1   4 0.5102564 0.6746032 0.6842105
## 5       1   5 0.5982906 0.5793651 0.5515351
## 6       1   6 0.3581197 0.6746032 0.4825911
## 7       1   7 0.0000000 0.6269841 0.5457762
## 8       1   8 0.4256410 0.6269841 0.4979621
```

```
## 9       1   9 0.5013736 0.7777778 0.6342105
## 10      1  10 0.6011396 0.6746032 0.4654971
## :       :   :         :         :         :
## 40523 310 414 0.4972291 0.6666667 0.5158263
## 40524 310 415 0.5778143 0.6746032 0.5065359
## 40525 310 416 0.4426564 0.6666667 0.4294118
## 40526 310 417 0.5315404 0.7152778 0.7070387
## 40527 310 418 0.5271102 0.6111111 0.7135914
## 40528 310 419 0.5204981 0.6944444 0.5683007
## 40529 310 420 0.5635103 0.5833333 0.4928843
## 40530 310 421 0.4891899 0.6111111 0.6108883
## 40531 310 422 0.6204433 0.6746032 0.7774510
## 40532 310 423 0.4233716 0.6746032 0.7908497
```

Linkage process

1. Clean the datasets
2. Generate pairs of records
3. Compare and separate columns of each pair
4. Score pairs using summing and probability
5. Select pairs that are matches based on theis score
6. Link the datasets together

```
# Create pairs
#pair_blocking(zagat, fodors, blocking_var = "city") %>%
  # Compare pairs
#  compare_pairs(by = "name", default_comparator = jaro_winkler()) %>%
  # Score pairs
#  score_problink()
```