RPersonalFinance

Intro

I wrote this program to help me learn R and to help me manage my personal finances. So far it has done a good job of both! In this document you will find an overview of the main functions of this program. IE how to use it. If you are interested in the development side, I welcome you to dig through the code in the GitHub repository here.

Premise

This program helps the user manage their personal finances across many accounts by taking a set of simple budget data and predicting account balances into the future.

Format

Data is stored in three different CSVs, each for a different purpose:

bills.csv keeps track of regular expenses for each bank account. *transfers.csv* keeps track of transfers occurring between bank accounts.

These are recorded by the day of month or day of week along with the total monthly amount.

special.csv keeps track of one-time transactions that occur on a specific date. This is useful to input a starting balance for a given month, or noting an annually-recurring bill.

Getting started

Let's start by importing the data and seeing what it looks like.

```
transaction_sheet <- import_data('data/bills.csv','data/transfers.csv')
special_sheet <- import_special('data/special.csv')
```

You're welcome to investigate these data frames yourself to see what's inside. They are more or less the same as the CSVs, which is to say the budget lines for a bunch of bank accounts. We can review one account at a time by filtering:

```
filter(transaction_sheet,bank_account=="Alice Primary")
```

```
##     bank_account accountor day               name monthly_amount
## 1  Alice Primary     Alice   5      Credit Card A            -77
## 2  Alice Primary     Alice  11      Credit Card B            -30
## 3  Alice Primary     Alice  20      Credit Card C            -30
## 4  Alice Primary     Alice  22           Car Loan           -280
```

```
## 5  Alice Primary     Alice  29    Credit Card Bill           -682
## 6  Alice Primary     Alice  22  Joint Contribution           -650
## 7  Alice Primary     Alice   7  Joint Contribution           -650
## 8  Alice Primary     Alice   1 Coffee and Shopping           -500
## 9  Alice Primary     Alice  15            Paycheck           1600
## 10 Alice Primary     Alice  30            Paycheck           1600
```

```
filter(transaction_sheet,bank_account=="Bob Primary")
```

```
##   bank_account accountor     day               name monthly_amount
## 1  Bob Primary       Bob  Friday   Credit Card Bill           -280
## 2  Bob Primary       Bob Tuesday Joint Contribution           -900
## 3  Bob Primary       Bob  Monday           Paycheck           1200
```

Note that Bob's budgets are scheduled by days of the week. Alice's in comparison are scheduled by day of month. This will be important in a moment.

Balance Sheets

Now that we've imported our budgets, let's turn them into a a set of balance sheets. A balance sheet looks like your bank statement. Here you'll see the sum of transactions on each day of the month with a running tally at the end of the day on the side.

create_balance_sheet()

Given a transaction sheet like the one we created earlier, create_balance_sheet() will create a set of balance sheets. It will automatically create one sheet for each bank account listed in the transaction sheet. It will create a balance sheet for as large a date range as your specify.

Note: dates are given in Lubridate format.

```
from_date <- mdy('11-01-2022')
to_date <- mdy('12-1-2022')

balance_sheet <- create_balance_sheet(transaction_sheet,from_date,to_date)
glimpse(balance_sheet)
```

```
## Rows: 251
## Columns: 4
## Groups: bank_account [8]
## $ date         <date> 2022-11-01, 2022-11-02, 2022-11-03, 2022-11-04, 2022-11-~
## $ amount       <dbl> -790, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -51, 0, 0~
## $ bank_account <chr> "Alice Credit", "Alice Credit", "Alice Credit", "Alice Cr~
## $ balance      <dbl> -790, -790, -790, -790, -790, -790, -790, -790, -790, -79~
```

There are 251 rows. That's 8 accounts * 31 days each. Let's filter out to the accounts we care about right now.

```
alice <- filter(balance_sheet,bank_account=="Alice Primary")
alice
```

```
## # A tibble: 32 x 4
## # Groups:   bank_account [1]
##    date       amount bank_account  balance
##    <date>      <dbl> <chr>           <dbl>
##  1 2022-11-01    929 Alice Primary     929
##  2 2022-11-02      0 Alice Primary     929
##  3 2022-11-03      0 Alice Primary     929
##  4 2022-11-04      0 Alice Primary     929
##  5 2022-11-05    -77 Alice Primary     852
##  6 2022-11-06      0 Alice Primary     852
##  7 2022-11-07   -650 Alice Primary     202
##  8 2022-11-08      0 Alice Primary     202
##  9 2022-11-09      0 Alice Primary     202
## 10 2022-11-10      0 Alice Primary     202
## # ... with 22 more rows
```

```
bob <- filter(balance_sheet,bank_account=="Bob Primary")
bob
```

```
## # A tibble: 32 x 4
## # Groups:   bank_account [1]
##    date       amount bank_account balance
##    <date>      <dbl> <chr>          <dbl>
##  1 2022-11-01   -107 Bob Primary     -107
##  2 2022-11-02      0 Bob Primary     -107
##  3 2022-11-03      0 Bob Primary     -107
##  4 2022-11-04    -70 Bob Primary     -177
##  5 2022-11-05      0 Bob Primary     -177
##  6 2022-11-06      0 Bob Primary     -177
##  7 2022-11-07    300 Bob Primary      123
##  8 2022-11-08   -225 Bob Primary     -102
##  9 2022-11-09      0 Bob Primary     -102
## 10 2022-11-10      0 Bob Primary     -102
## # ... with 22 more rows
```

Weekday transactions

Take a close look at Bob's account. Notice how there's a few repeating transactions, such as the -70 on 11/11, 11/18, 11/25, and so on. Notice also that $70 = 280/4$, and 280 is the monthly value of Bob's Credit Card Payment which recurs on Fridays.

If a bill is listed as occurring on a named weekday, it will be automatically split up onto all of that weekday in the month, at 1/4 of the monthly value.

Predicting overdrafts

predict_max_overdraft(balance_sheet)

This function is very useful. It will find the date and value of the balance furthest below 0 for each account in the sheet provided. See for yourself:

```
joint <- filter(balance_sheet,bank_account=="Joint Primary")
predict_max_overdraft(joint)
```

```
##         date balance  bank_account
## 1 2022-11-06    -292 Joint Primary
```

Keep in mind this doesn't tell us when the first day the account will be overdrawn: But it does tell us that we need to move $517 in total to avoid a negative balance.

Visualizing

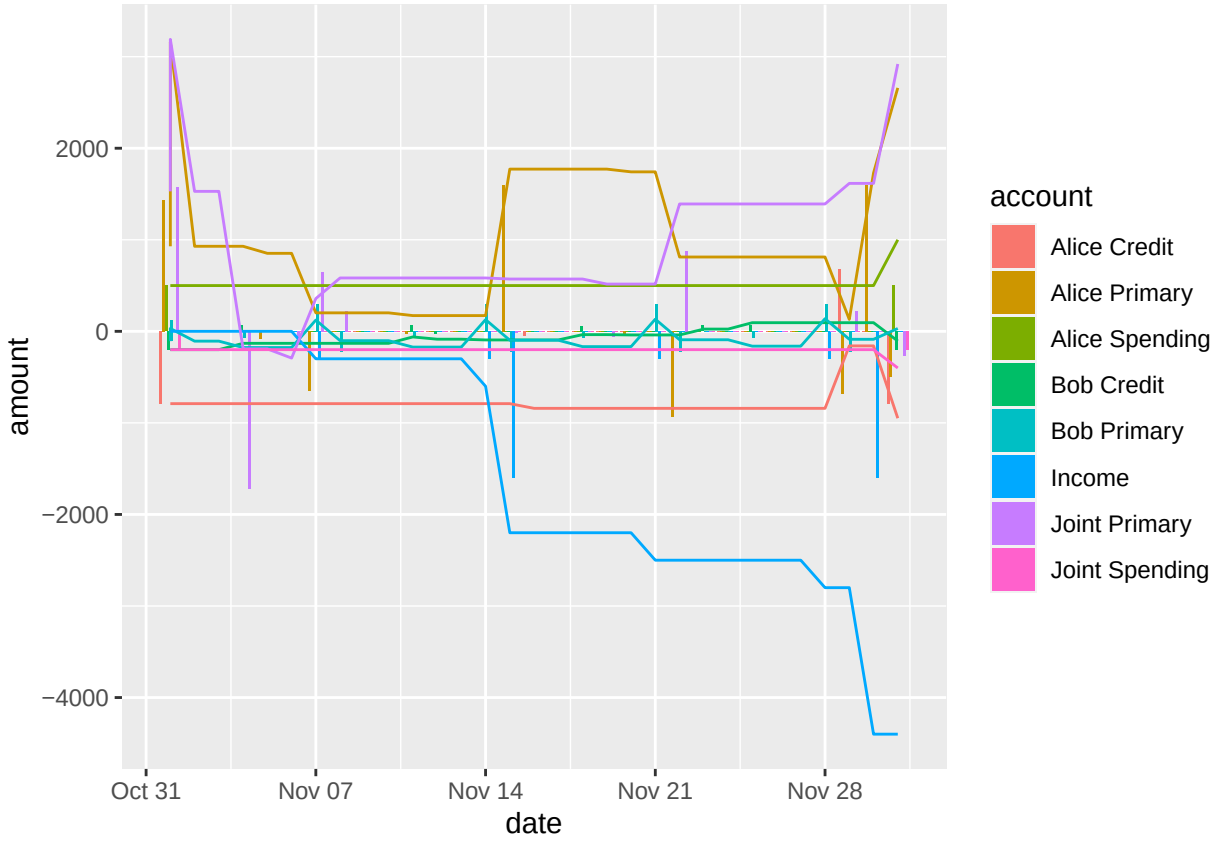
draw_balance_sheet()

Finally, let's visualize the data.

By default this function displays transactions and overall balance in a chart, separated into different color groups by bank account. It's a combination chart, with a line part representing the running balance and bars representing total transactions on a given day.

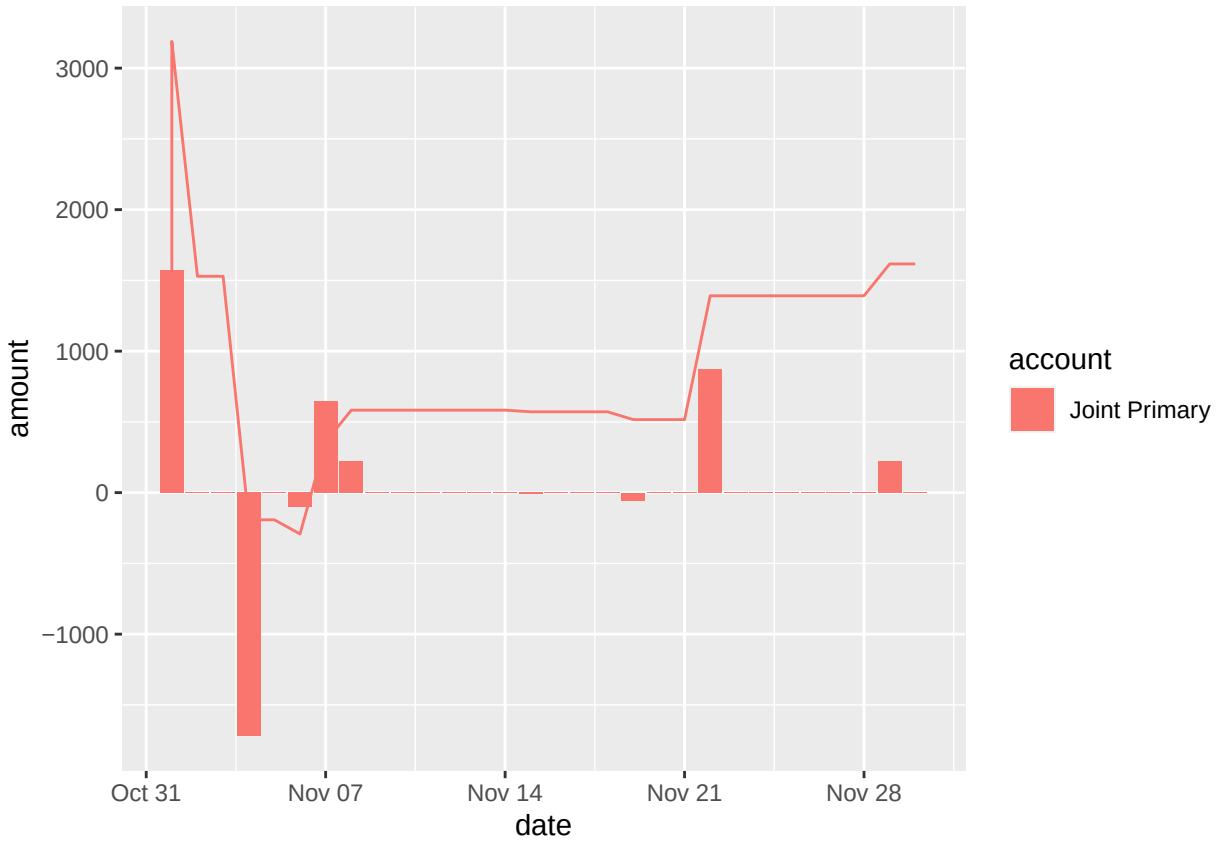
It will display as many accounts for as wide a range of dates as you supply it, which can get messy fast.

```
draw_balance_sheet(balance_sheet)
```



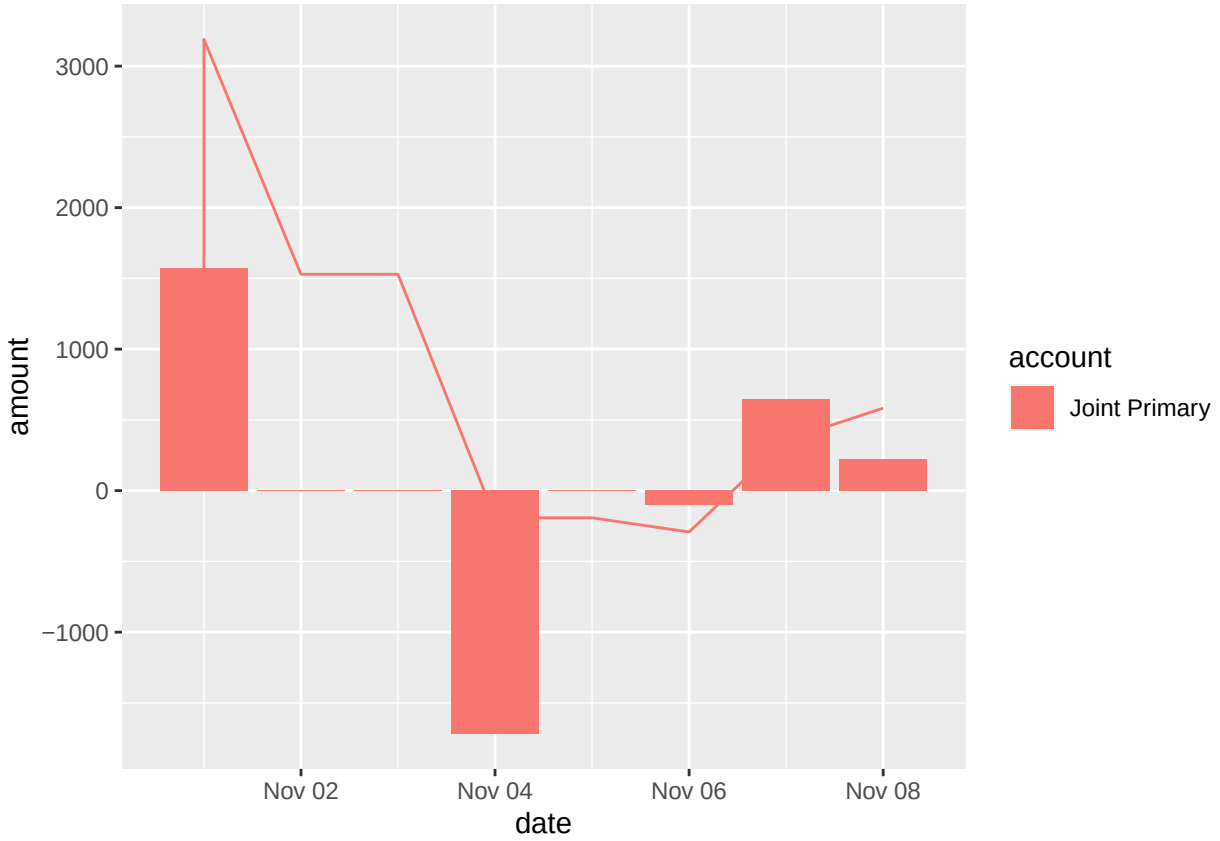
So we might want to filter it down a little.

```
joint <- filter(balance_sheet,
                bank_account=='Joint Primary',
                between(date,mdy('11-01-22'),mdy('11-30-22'))
                )
draw_balance_sheet(joint)
```



By the way, we can use this to analyze that overdraft we found out about earlier. Let's zoom in to just that week.

```
joint <- filter(balance_sheet,
                bank_account=='Joint Primary',
                between(date,mdy('11-01-22'),mdy('11-8-22'))
                )
draw_balance_sheet(joint)
```



So here we see what our account looks like around the date of the most negative balance. Looks like there's a large deduction that day. We can get more details about that time range this way:

```
overdraft <- transaction_sheet %>%
              filter(
                     bank_account=="Joint Primary",
                     between(day,1,7)
                    ) %>%
              transmute(day,name,monthly_amount)
overdraft
```

Yep, so it looks like we are a little short for rent this month. But we know by exactly how much, so as long as we move $518 into Joint Primary by 11/4, we should be able to cover it!