

Lab 4: Data Transformations

Overview: Large Technology Stocks

For this assignment we practice data transformation using a dataset of the daily prices and daily trading volumes of a group of large technology stocks that trade on US stock exchanges. [Click here to download stocks.csv](#), which contains data going back to 2000. The dataset contains several variables, including:

- symbol: which is the ticker symbol for the stock
- date: which is the trading date
- open, high, low, and close, which are the price at the start of trading, the high price during the day, the low price during the day, and the stock price at the close of trading (unit is USD)
- adjusted: which is the stock price at close adjusted for the financial effects of special events (such as dividends). Unit is USD
- volume: which is the number of shares which traded during a given trading day.

```
library(tidyverse)
library(TTR)
library(skimr)
library(kableExtra)
stocks = read_csv("/home/georgehagstrom/work/Teaching/DATA607/website/assignments/labs/labData/stocks.csv")
diabetes = read_csv("/home/georgehagstrom/work/Teaching/DATA607/website/assignments/labs/labData/diabetes.csv")
skimmed = diabetes |> skim()

skimmed[c(2,3:6)] |> print()
```

```
# A tibble: 9 x 5
  skim_variable      n_missing complete_rate numeric.mean numeric.sd
  <chr>             <int>         <dbl>         <dbl>         <dbl>
1 Pregnancies         0             1             3.85          3.37
```









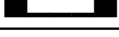
2	Glucose	0	1	121.	32.0
3	BloodPressure	0	1	69.1	19.4
4	SkinThickness	0	1	20.5	16.0
5	Insulin	0	1	79.8	115.
6	BMI	0	1	32.0	7.88
7	DiabetesPedigreeFunction	0	1	0.472	0.331
8	Age	0	1	33.2	11.8
9	Outcome	0	1	0.349	0.477

```
skimmed[c(2,7:11)] |> kable()
```

skim_variable	numeric.p0	numeric.p25	numeric.p50	numeric.p75	numeric.p100
Pregnancies	0.000	1.00000	3.0000	6.00000	17.00
Glucose	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	0.000	0.00000	23.0000	32.00000	99.00
Insulin	0.000	0.00000	30.5000	127.25000	846.00
BMI	0.000	27.30000	32.0000	36.60000	67.10
DiabetesPedigreeFunction	0.078	0.24375	0.3725	0.62625	2.42
Age	21.000	24.00000	29.0000	41.00000	81.00
Outcome	0.000	0.00000	0.0000	1.00000	1.00

```
skimmed[c(2,12)] |> print() |> kable()
```

```
# A tibble: 9 x 2
  skim_variable      numeric.hist
  <chr>             <chr>
1 Pregnancies
2 Glucose
3 BloodPressure
4 SkinThickness
5 Insulin
6 BMI
7 DiabetesPedigreeFunction
8 Age
9 Outcome
```

skim_variable	numeric.hist
Pregnancies	
Glucose	
BloodPressure	
SkinThickness	
Insulin	
BMI	
DiabetesPedigreeFunction	
Age	
Outcome	

All of the the stock prices and the trading volume have been adjusted for stock splits, so that the data provide a continuous record of how prices and trading volume changed.

There are several important functions and packages that you will need to use to complete this exercise.

- We will make a lot of use of window functions in `dplyr`, which are very helpful for making transformations (including `lead`, `lag`, `percent_rank`). The [dplyr vignettes and articles are incredibly useful for learning about all the different functions available](#)
- We will use the `TTR` package (which is part of the `tidyquant` family of packages, [see here](#)). The main function we will use is called `runMean`. An alternative package that is also very nice but not part of the `tidyverse` is `RcppRoll`
- If you aren't very comfortable with logarithms, you should read more about them. They are one of the most important mathematical functions for data science. We aren't using their mathematical properties much this week but they will be important throughout your data science journey. [Khan Academy has a decent video](#), and [this article in the journal nature](#) has some more context.
- We will calculate some correlation coefficients, using the `cor` function from base R (`?cor` to see how it is used). There is also a `tidyverse` package called `corrr` that is useful for calculating correlations on data frames, but we won't use it for this lab.
- The motivation for today's assignment came from some news articles a few years ago about how big tech stocks collectively had a miniature meltdown after powering the stock market for several consecutive years, see [this article at Morningstar](#)
- The [wikipedia page on Market Impact](#) has some references to Kyle's λ , which we will calculate this week.

Problem 1: The price of a stock on a given day only conveys information in relation to the stock price on other days. One useful measure is the daily `return` of the stock, which we will define as the ratio of the adjusted closing price on the current day of trading to the adjusted closing price on the previous day of trading. Read the following article on *window functions* in `dplyr`: [window functions in dplyr](#). Find a function there that will help you calculate the daily return and use it along with `mutate` to add a `return` column to the data frame containing the daily return. Hint: make sure to use `group_by(symbol)`, otherwise your calculation might transpose prices from a different stock at the beginning of each time series.

Differences between the adjusted return and the return measured on the close price should indicate special corporate events such as dividends. Calculate the un-adjusted return using the same technique you used to calculate the return, but replacing the `adjusted` variable with the `close` variable, and find the datapoint in the dataset where the return exceeded the unadjusted return by the greatest margin. (Hint to check you have done it right: it happened in November 2004).

If you are curious: Look for an old news article describing the significance of that event and tell me what happened

Problem 2: When working with stock price fluctuations or other processes where a quantity increases or decreases according to some multiplicative process like a growth rate (for example population growth) it is often better to work with the `log` of the growth rate rather than the growth rate itself. This allows standard summary statistics such as the mean to have a useful interpretation (otherwise you would have to use the geometric mean). Furthermore, the `log` transform is often useful to use on variables that are strictly positive, such as population growth rates or daily stock returns. To see why, consider a hypothetical stock which had a return of 0.5 (50% loss) on one day and 1.8 on the next day (80% gain). The mean of these two returns would be 1.075, or 7.5% per day. However, at the end of the two day period the stock would have lost 10% of its value ($0.5 \times 1.8 = 0.9$). If we had computed the mean of the `log(return)` instead, we would have found that $(\log(0.5) + \log(1.8))/2 = \log(0.9^{(1/2)})$, or approximately -5.2% per day, matching the observed price change.

Create a new variable called `log_return` which is the `log` of the return variable you calculated in the previous problem. Generate either a histogram or density plot of the distribution of `log_return` for the entire dataset. Then create a QQ-plot of `log_return` using `geom_qq()` and `geom_qq_line()`. What do you notice about the “tails” (right and left side/extreme edges) of the distribution from the QQ-plot? Are there visible signs of this in the density plot/histogram that you made?

Problem 3: Volume measures how many shares were traded in a given stock over a set time period, and high volume days often associate with important events or market dynamics. Make a scatter plot of `volume` versus `log_return`, faceted by

symbol to account for the fact that different stocks have different trading volumes. Do you see an association between volume and log_return in these scatter plots?

Use the `cor` function to compute the pearson's correlation coefficient between `volume` and `log_return` for each symbol. Why do you think the correlations are close to 0? (Hint: use it with `summarize` and don't forget that `cor` is a base R function so you will either need to filter NA values for volume and log_return or appropriately choose the `use` flag in the argument- see `?cor` for more info).

Next compute the correlation in the same manner but this time transform `log_return` using the absolute value function. Recreate the faceted scatter-plots from the first part of the problem but with the absolute-value transformed `log_return`. How have the correlations changed from the previous summary?

Problem 4: For this problem we will implement a more complicated mathematical transformation of data by calculating a measure of liquidity for each stock.

Liquidity is defined loosely as the ability for a given asset to be bought or sold without a large impact on price. Liquid assets can be bought and sold quickly and easily, whereas illiquid assets have large increases or decreases in their price when someone tries to buy or sell them in large quantities. Liquidity is considered an important property of a well functioning financial market, and declines in liquidity have been blamed for worsening or triggering stock market crashes.

Many methods have been invented to measure liquidity, but for this problem we will focus on one a method called "Kyle's λ ". Kyle's λ estimates liquidity by using a linear regression between the absolute value daily return of a stock and the logarithm of the dollar volume of that stock. The time periods used to estimate this regression can vary, but here we will use daily returns and a one month (defined as 20 trading days). You will learn a lot about linear models in DATA 606 and other classes, but to be complete, λ is a coefficient in the following linear model:

$$|R_t - 1| = c + \lambda \log((\text{Volume})_t(\text{close})_t) + \epsilon_t$$

where the coefficients c and λ will be calculated to minimize the error ϵ_t over the past 20 trading days.

λ stands for the amount that the stock price will move in units of basis points for a given log dollar volume of trade. A small λ indicates high liquidity, and a high λ indicates low liquidity.

λ can be calculated using rolling averages on the time series data with the `TTR` package, specifically the function `runMean` which when used within a `dplyr` pipeline will calculate the mean over the past n data points. For example, the command:

```
library(TTR)
stocks |> group_by(symbol) |> mutate(log_return_20d = runMean(log_return,n=20))
```

adds a new variable which is equal to the mean of the log_return over the past 20 days. The mathematical formula for λ is:

$$\lambda = \frac{\text{mean}(R_a \log(p_c V)/20 - \text{mean}(R_a) \text{mean}(\log(p_c V)))}{\text{mean}(\log(p_c V)^2)/20.0 - \text{mean}(p_c V)^2}$$

where to make the formula easier to read we have defined $R_a = |\text{return} - 1|$, $p_c = \text{close}$ and $V = \text{volume}$, and the averages have been taken over the past 20 days of data.

Add a new variable called `kyle` to the data frame by implementing the above formula for λ . Make sure to read and implement the formula very carefully, and to use the `runMean` function to calculate the rolling average correctly.

Plot Kyle's lambda for each stock over time (I would use a faceted scatterplot). What do you notice about how this measure of liquidity behaves (remember liquidity is high when λ is small)?

Next add a new variable to the dataframe called `extreme` which is true when the log_return for a given stock is *either* greater than 95% of other values of the log_return *or* less than 95% of all values of log_return. Use the `percent_rank` function along with logical operators to create this variable. Then for each stock calculate the mean value of Kyle's lambda for the days when the log_return had extreme values and for when it didn't (as identified by the `extreme` variable). What do your calculations and figures indicate about liquidity during extreme events?