Overview over the semester, setting up your computer, and first R program

Yvan Lengwiler
Faculty of Business and Economics
University of Basel

yvan.lengwiler@unibas.ch

File: prep-01.tex

Contents

Part I

IT stuff

As mentioned in the syllabus, we will install some software together that we will need for this course. We will seriously begin studying Finance next week. In this introductory session, we will prepare our computers so that we can exchange files efficiently and are ready to create some useful code.

We will use `git`, a version control system, and `Github`, a public place to store version-controlled projects, in this course. For this reason, you have to have `git` installed on your machine and have a `Github` account. So it a good idea to get used to it; that way you can hit the floor running. [1] Honestly, we could do it differently, but it is likely that you will encounter `Github` and `git` in your professional life.

By the way: I learned the basics of git and Github from Nick White's video at `https://youtu.be/mJ-qvsxPHpY?si=zcRIjqRiU_PJ6uXE`. I highly recommend to watch this 20 minues video.

1 Install git

If you have a Mac or use some Linux, you should already have git installed.[2] If not, you can easily install it. For Windows, you normally have to download and install it and the process is a little more involved.

We start by installing the git software on our local machine,

Installing git **Windows**

Browse to https://github.com/git-for-windows/git/releases/ or https://git-scm.com/download/win and download the 64-bit version. You should check that the downloaded file is virus-free. You can do that with your local antivirus software. Alternatively, you can upload the downloaded file to https://www.virustotal.com/ and let this website check your file using multiple virus detection packages. If it is clean, install it on your Windows machine.[3]

[1] A more in-depth explanation of how the two interact, and how exactly to use the version-control feature of git (meaning, you can go back to a previous version of your work) is provided in the separate handout, see the file `'git.pdf'` on `ADAM` and on Github. We will not go into any depth about this here.

[2] Check with `'which git'`.

[3] The Linux and Mac versions simply provide access to the `git` program from the command line. The Windows version does that, too, i.e. you can use the `git` from the command terminal. But in addition, the Windows installation also provides a new type of shell, called `bash`. This is accessible when you right-click in some folder. You can use `git` from the `bash` shell in Windows, which allows you to use the Linux syntax. Essentially, this means that you can also use the Linux commands descrobed below, provided you use the

Installing git

Linux

```
Debian, Ubuntu:  # apt install git
RHEL based:  # dnf install git
Arch:  # pacman -S git
SUSE:  # zypper install git
```

MacIntosh

```
$ brew install git
```

Before we use git, we have to introduce ourselves with our name and email address,

Global config Linux, MacIntosh, Windows

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

2 Downloading a local copy of my repo for class

The git system that is now on your machine is a version control system. It allows you to manage different branches of a project that you develop, and go back to previous versions. How you do that exactly is not part of this class. Check out the seperate handhout `'git.pdf'` to learn more about this, or study the documentation.

A *git server* is not the same as git itself. It is a place where git repositories can be stored and accessed over the internet. `https://github.com` and `https://gitlab.com` are the most important public git servers. A public git server has the advantage, that you can share your files with everyone. Likewise, you can download files from other people who have shared their :files easily over the internet using such a public git server.

For this class, I have prepared a publicly available repository on my Github account. You can pull these files down to your local computer. Likewise, you will push your own work to your own Github account during the semester, but we will come to that later.

Browse to `https://github.com/yvan-lengwiler/Finance-Uni`. This is the repo I have prepared for you. You can easily download a copy of this by cloning it. First make a directory on your computer that you will use for this class. I will assume that you call this directory `'FinMaster'`, but you are free to name it whatever you like.

Windows `bash` shell instead of the standard Windows command shell. There is also a GUI for `git`, which is not described in this handout.

Cloning my repository Linux, MacIntosh, Windows

```
mkdir FinMaster
cd FinMaster
git clone --depth 1 https://github.com/yvan-lengwiler/Finance-Uni.git
cd Finance-Uni
```

The `'git clone'` command produces a sub-folder with the name of the repo you have cloned, here `'Finance-Uni'`. This folder contains subfolders `'week1'`, `'week2'` etc. I will fill these as the term progresses.

3 Make your own account on Github

You can only download from by Github repo, you cannot upload to it. Instead, you will make your own Github account so that you can upload material yourself.

Browse to `https://github.com`. Sign up for an account. If you already have one, you can use that, or you can make a new one just for university work — it's up to you.

Note that Github has allowed for two-factor authentication (2FA) for years. They have begun requiring it. If you are active on your account, it is likely that at some point you will be notified by the website that you have to provide 2FA. You can use an authentication app (such as Authy or the like), a physical authentication device (like Ubikey), or SMS. SMS is considered the least secure because nothing is encrypted in SMS and SIM cards can be spoofed. In any case, it is advisable to install two types of 2FA because that way you can still access your account even if one method does not work (for instance, because your smartphone died).

3.1 *Authentication part 1: making an SSH key pair*

Github will not allow you to upload files to your repo just with a password (even with 2FA enabled). A more secure auhentication method is requires. One possibility is to access Github through the "secure shell protocol" SSH. For that, we need to create SSH key pairs. A key pair consists of a private key (akin to an actual key) and a public key (akin to a lock). You can share the public key file; you should never share the private key file.

You can create an SSH key pair like this,

SSH key pair Linux, MacIntosh, Windows

```
ssh-keygen -t ed25519 -C "github:unibas-account"
```

The command will ask for a name of the files. The default is `"id-ed25519"`. You might already have files with that name that you probably want to keep, so you should

choose another name. Since we create these explicitly for Github, we could use the name `"github-ed25519"`, for instance.

The dialog will also give you the opportunity to use a passphrase for the key. Choosing a passphrase means that the key can only be used by a person who knows the passphrase. If someone steals your private key but does not know the passphrase, the key is useless.

The command produces two files, `github-ed25519` and `github-ed25519.pub`, or whatever the name of the file you chose. The `.pub` file is the public key, the other one is the private key. Make shure to put there files in the `$HOME/.ssh` (Linux, Mac) or `"C:\Users\YOUR-USERNAME\.ssh"` (Windows) directory. It will not work otherwise. If this directory does not exist, create it.

3.2 *Authentication part 2: storing your public SSH key in your Github account*

Now, view the content of the **public key** on the screen, with `'cat github-ed25519.pub'` (Linux, Mac) or `'type github-ed25519.pub'` (Windows). Mark it with your mouse and copy the content (this should be just one line) to the clipboard.

Now, go back to the Github website. In the main menu, go to `"settings"`, then `"SSH and GPG keys"`, then click on the `"New SSH Key"`. Give the key a descriptive name that allows you to later understand which key this belongs to, choose `"Authentication key"`, and paste the content of your new **public key** into the form. *Never ever paste a private key in there!*

3.3 *Authentication part 3: telling local git to use your key*

Back in the terminal, you need to help your local git installation find the correct SSH key when pushing to Github. The easiest way to achieve this is by configuring this globally.[4]

Go to the `"$HOME/.ssh"` directory (in Linux or Mac) or `"C:\Users\YOUR-USERNAME\.ssh"` (in Windows), respectively. The new public and private key files should be there. Create an additional file called `"config"` in this location. If you already have such a file, just add the content below to it.

```
Host github.com
    Hostname github.com
    User git
    IdentityFile ~/.ssh/github-ed25519
    AddKeysToAgent yes
```

The user and host name in the internet is specified in the `'Hostname github.com'` and `'User git'` lines and this must be entered exactly as shown here. `'github-ed25519'` (the name of the `'IdentityFile'`) should be substituted for the name of your key. This config

[4] Be careful: If you have multiple accounts at Github, this will mess things up. Checkout the guide `'git.pdf'` for ways to handle this sutuation.

file will make sure that whenever we try to reach `git@github.com` through SSH, your new SSH key will be used to authenticate.

You can check if this works with the command `'ssh git@github.com'`. This should produce an output similar to this,

```
---------------------------------------------------------------------------
'PTY allocation request failed on channel 0
Hi yvan-lengwiler! You've successfully authenticated, but GitHub does not
provide shell access.
Connection to github.com closed.
---------------------------------------------------------------------------
```

This is, in fact, an error message, but it is the message you want to get. The connection was established but then was closed because Github does not allow access through a shell. If the output mentions `"connection refused"`, then it did not work.

4 Preparations for class

Just as I provide files for you to download via Github, you will upload your work to your personal Github repositories. Setting this up correctly is a little bit of work.

4.1 *Make a new, empty, online repo*

Launch your browser, go to the Github website, and log into your account. Under `"Repositories"`, click the `"NEW"` button, give the repository your student number as the name. Fill out the rest of the form. Make sure that you create a `'private'` repo, not a `'public'` one. For the moment, do not create an automatic `'README'` file and do not choose a license.[5] Click `"OK"`.

4.2 *Make a local repo*

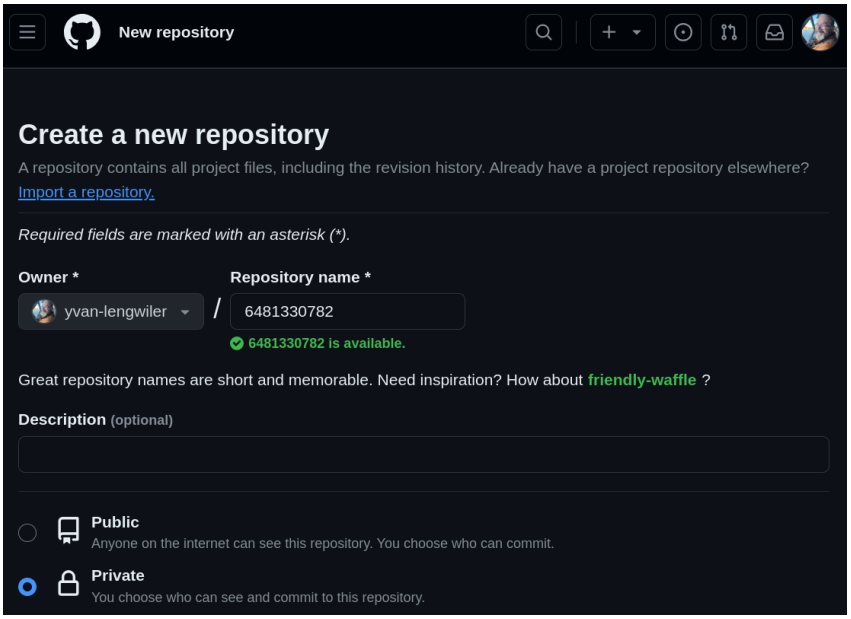
Back in the terminal (not in the browser), go to your `'FinMaster'` folder and create a new, empty subfolder that has your student number as name:

Initialize repository Linux, MacIntosh, Windows

```
mkdir MATR_NR   (replace MATR_NR with your student number)
cd MATR_NR
git init -b main
```

The last command initializes this directory as a git repository. Now, put some files into this folder. For instance, copy the relevant material from my repo to your newly created folder.

[5] Choosing a license is extremely important if you make a public repo to protect yourself against liability claims!



Copy files into your own working directory

Linux, MacIntosh

```
$ cd FinMaster/MATR_NR
$ cp -r ../Finance-Uni/* .
$ ls -al
```

Windows

```
> cd FinMaster/MATR_NR
> robocopy /MIR ../Finance-Uni/* .
> dir
```

If all worked well, you should have six subfolders now from `week1` to `week6`. These are the files you will work on during the semester.

4.3 *Commit to the local repo*

The next step is to syncronize these locally stored files with your Github account. Make sure that your current firectory is `FinMaster/MATR_NR`. The issue the following commands:

Commit to repository Linux, MacIntosh, Windows

```
git add .
git commit -m 'first commit'
```

You can check if this has worked with `'git log'`. This should produce a feedback like this

```
------------------------------------------------------------------------
commit 67c11ca163eb156e0d9e7b0c631618177691c94d (HEAD -> main)
Author: Yvan Lengwiler <yvan.lengwiler@unibas.ch>
Date:   Sun Mar 24 16:04:00 2024 +0100

    first commit

------------------------------------------------------------------------
```

4.4 *Push your local repo to Github*

Now you can push this local repository from your drive to Github. In the terminal, while still in your `'MATR_NR'` folder, do as shown in the next box.

Pushing local repository for the first time Linux, MacIntosh, Windows

```
git remote add origin git@github.com:YOUR-GITHUB-ACCT-NAME/MATR_NR.git
git push -u origin main
```

Let us unpack this. The first command creates a connection between your local repo (the folder you are currently operating from in you computer) to your Github account. The second line informs git which key to use when authenticating to Github. The third line, finally, pushes your local repository to the Github website.

If everthing has worked, a copy of your local directory should now be online in your Github account. To check, browse to Github, log into your account, and see whether your `'MATR_NR'` repository has been populated.

This somewhat complicated sequence is necessary only the first time. Pushing new versions to your Github account is much easier. If you have changed or added files in the `FinMaster/MATR_NR` folder (or in some subfolders), you can first add and commit them to your local repo with

```
git add .
git commit -m "descriptive comment"
```

and then push them to Github with

```
git push
```

4.5 *Give your teachers access to your private repository*

Your Github repository is private, meaning it is not visible for random visitors. However, your teachers will need access to it to see what you do. You shoud tehrefore give your lecturers access to your private repository. My account name is `"yvan-lengwiler"` and that of the

teaching assistant is not defined yet (because the teaching assistant is not determined yet). Add us on to your newly created Github repository like this: `Settings > Collaborators > Manage Access > Add people`. You can revoke this access after the semester.

5 Workflow during the semester

During the semester, I will update my files in the `"Finance-Uni"` repo. You can pull down the latest version by going to the `"Finance-Uni"` directory on your local computer and issue the command

```
git pull
```

This should download all the changes I made on my side to your computer. You can then copy the files for the current week from `FinMaster/Finance-Uni/week...` to your working directory `FinMaster/MATR_NR/week...`.

When you then prepare for the lecture or solve homework, you will work within the `"MATR_NR"` folder and the subdirectories in there, `"week1", "week2", ...`. You will push your work to Github with the

```
git add .
git commit -m "descriptive message"
git push
```

sequence when you are ready.

6 A final word about git and the requirements of the labor market

There is much more to git. To understand it better and use it more fully, you can study the separate handout `"git.pdf"`. This is still superficial and `'git'` has several other functions that are not covered in the handout. The official documentation is available at `https://git-scm.com/doc`. `git` and the `R` language are only tools we use. They are not the stars of this show. Our real topic is, of course, **FINANCE**.

It is, however, a fact, that most finance jobs today cannot be neatly separated from computer science. Even if you will not be a code developer, people working professionally in finance today are expected to code at least to some extent, and that also means sharing code and contributing to the codebase of your employer. In most modern finance houses, this means R, Python, C/C++, and, yes, git as well. (In other companies, it just means Excel :-)

Part II

First look at financial data

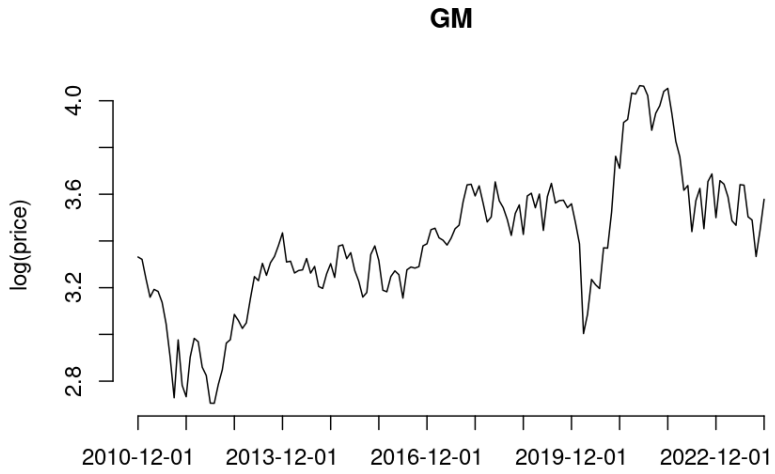
7 Exploring typical properties of an equity price

After all this IT business, we are now ready to start exploring Finance.

Equities are shares of ownership of companies and represent productive capital. They are the main investment vehicle that drives any capitalist economy. Equities of many large companies are publictly traded on organized exchanges and we we can thus easily track their market prices. Let us download a time series of a significant company from a website that provides such data for free (`finance.yahoo.com`) using the `R` programming language. Please study the supplied program `'explore-an-equity.r'` to see how I do this. Here are the most important parts of the code:

```
# *** parameters ************************************************************
symbol    <- 'GM'               # General Motors
interval  <- '1mo'              # 1d, 1wk, or 1mo
from_date <- '2010-01-01'
to_date   <- '2023-12-31'
# *** constructing the url **************************************************
basedate <- as.Date("1970-01-01")
fromcode <- difftime(as.Date(from_date), basedate, units="secs")
tocode   <- difftime(as.Date(to_date), basedate, units="secs")
url <- paste0('https://query1.finance.yahoo.com/v7/finance/download/',
    symbol, "?period1=", fromcode, "&period2=", tocode,
    "&interval=", interval, "&events=history&includeAdjustedClose=true")
# *** acquire data **********************************************************
data <- read.csv(url, header=TRUE)
price <- data$Adj.Close
dates <- data$Date
# *** compute returns *******************************************************
# annualized return rate from one observation to the next
yield <- diff(log(price)) * factor
mu <- mean(yield)       # mean return rate
sigma <- sd(yield)      # volatility
cat("mu =", round(100*mu,1), "%, sigma =", round(100*sigma,1), "%")
. . .
# *** compute density estimate and Q-Q plot *********************************
density_estimate <- density(yield, kernel = "epanechnikov")
. . .
qqnorm(yield)
```

The result is the chart below.



1 Question. Try to describe the most relevant features of this time series. Is it random? Is there a trend? etc.

Your answer here:

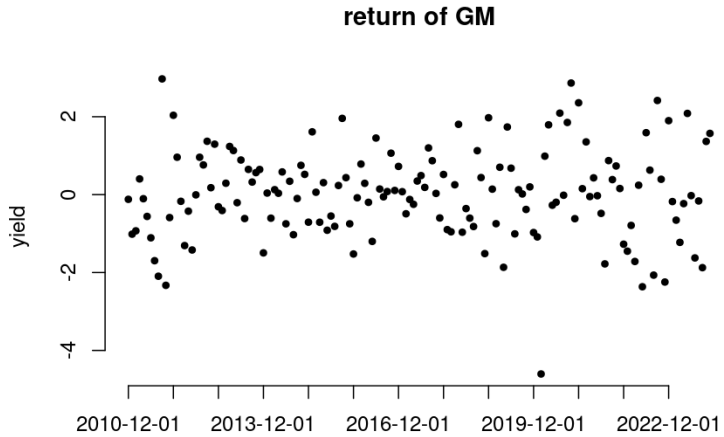
---

2 Question. The chart is represented on a logarithmic scale. What are the two main advantages of the log scale here?

Note: If you are interested in such things, check out `https://youtu.be/0jIwC0evUew?si=BtzwSN5DI6NdE6CN` for a discussion of the history of the logarithm.

Your answer here:

Instead of studying the level of the share price, we can also study the *yield* or *return rate* that this path entails. Doing this for our data produces



The stochastic properties of yield time series are central in finance, so it is worth to study this in some depth.

3 Question. What can you observe with the naked eye?

Your answer here:

The script also computes an estimate of the distribution of the returns (a so-called kernel estimate) and a Q-Q plot, which allows us to compare the empirical distribution with the normal distribution. We often work with the assumption of normally distributed returns in finance, so it is worthwile to check if this assumption is valid empirically. In your homework for next week, you will explore under what circumstances this appears to be justified, and what shape the deviations from the normal distribution take.

I would like you to play around with this script and explore the characteristics of the GM and other stocks at the 1-month and at the 1-day interval. We will discuss this next week. Please see the handout for next week about what you should do. There is some amount of programming involved, as well as a bit of reading.

8 Studying the R code

The preparation for next week will require you to do some programming. It is therefore useful for you to understand the `'explore-an-equity.r'` script. It consists of nine sections. The first is `'preparations'`. This first installs R libraries we require. It also sets the working direktory to the location of the script. The second section is `'parameters'`. He we simply set some parameters that determine what data exactly will be used. We choose here the `'symbol'` of the asset we download and the interval as well as the time period of the data to be used. The third section is called `'number of observations per year'`. Do you understand what we do here? The remaining sections are called

- `'constructing the url'`,
- `'acquire data'`,
- `'plot it'`,
- `'compute returns'`,
- and once more `'plot it'`,
- then `'computer density estimate and Q-Q plot'`.

We go through this code together. Try to understand each line here.

9 Ready for sharing in Zoom

Have this ready for the Zoom meeting. Prepare some slides with all your insights and thoughts and the problem solving you did in the learning boxes above. Have these slides ready to be shared in the Zoom meeting.