Quantitative Methods

Working Environnement and data base

Ludovic Vigneron

University of Lille (IAE)

September 6th, 2022

Morning program

1. Intalling Softwares
2. R langage basis
3. WRDS data base
4. Extracting/importing data

Intalling Softwares

R langage

R is a programing langage dedicated to statistics and data sciences.

- **easy to use** (simple synthax especially using the tidyverse)
- **free** (that makes a huge difference with Stat, SAS, Eview, SPSS...) and multi-platform (windows, OS, Linux)
- **evolutive** (a very important community develop applications -free packages- and makes 'R base' evolve on a regular basis)
- **a lot of information available** (R journal, R bigbook, R-BLOGGERS, Stack overflow, freakonometrics, MyEconTricks...)

R langage

You can download R at the following adress https://www.r-project.org/ .

R Studio IDE

- What is a *IDE* ?

It is just a *Interactif Developpement Environment*, a software that helps to deal with application creation using one or more programing language.

R Studio is dedicated to data analysis. It provides tools to get data, to shape data (data wrangling), to draw graphs, to do statistics (test, regression models, other modeling...) and to produce reports (presentations, text documents, interactive product...).

Actualy, R studio is becoming posit. It will give more space to other programing langage (like Pyhton, Julia,C++...).

R Studio IDE

You can download R Studio at the following adress https://www.rstudio.com/products/rstudio/download/ .

Video installation guide

To help you to install R and R Studio IDE, I invite you to watch the follow trought video (duration about 5 minutes) here after.

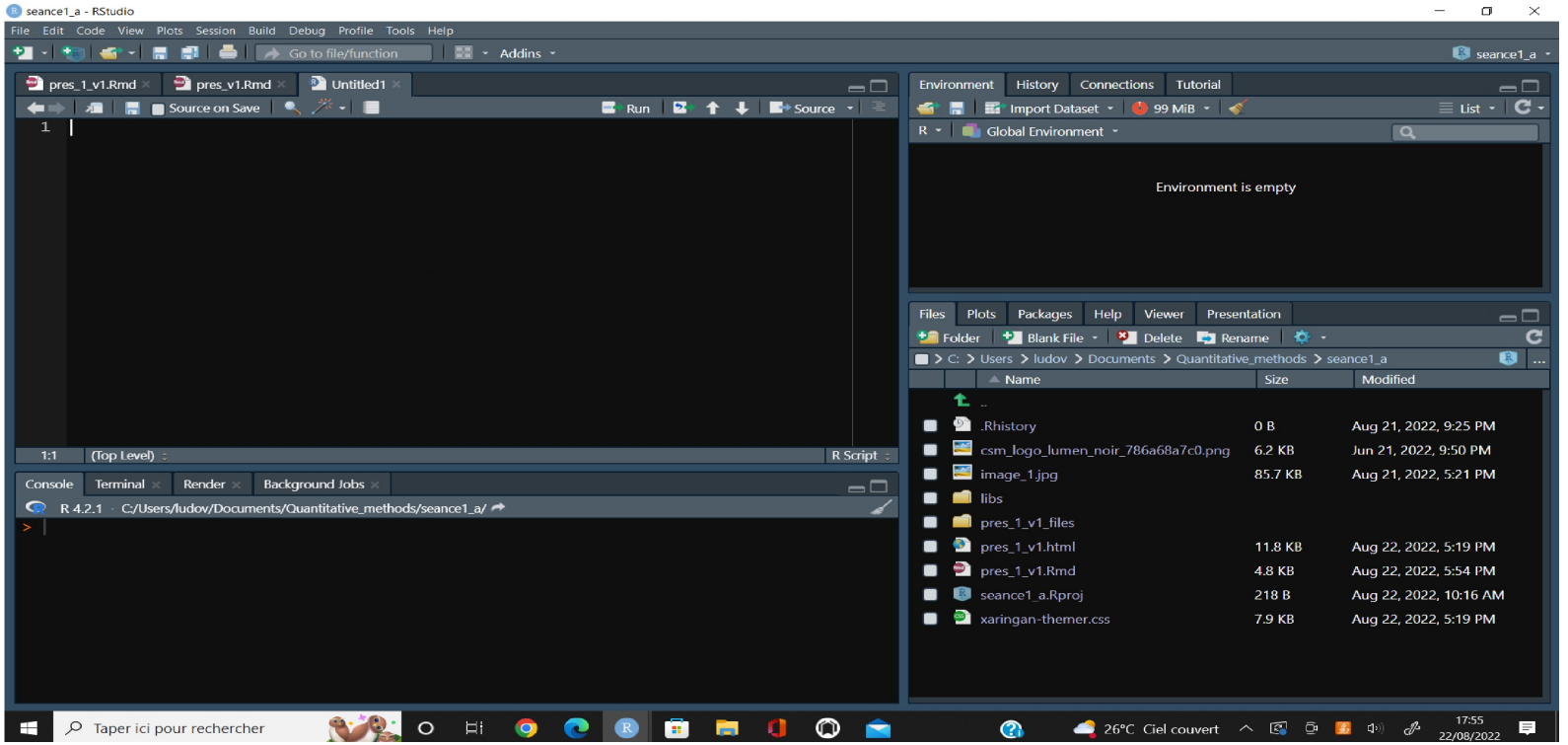
https://youtu.be/hT2tSvj7d1A

Presentation of the R Studio environement

You will find a short presentation of the R Studio in the following video (duration about 15 minutes).

https://youtu.be/fB5QlyjWqK0

Presentation of the R Studio environement



Presentation of the R Studio environement

Four panel layouts

- **Source** : where you write your program and save it to make your analysis reproductible.
- **Console** : where you interact with R.
- **Environment** : where you find the different objects that you have created during your session.
- **Files**, **Plots**, **Packages**, **Help**, **Viewer**, **Presentation** : you find the files in your working directory, you can view the graph made, the list of your installed packages, some descriptions of fonctions for which you have asked for help...

Presentation of the R Studio environement

It is important to create a new project when you start a data analysis.

file -> new project

Indicate where to put the directory associated with your project and name it.

If you have some doubt on your actual working directory type **getwd()** in the console space.

In order to make your work reproductible, you must at least use a script file.

file -> new file

Warning : it is not because you have saved your project (.Rproj) that your data and your programs (.R,.Rmd...) are also saved (.csv,.xlSx...). **You have to save them independly!!**

R langage basis

The operators

- math : +, -, *, /, %/%,

  `%*%, t(), ^, log(), exp()...`

- assignation : <- = ->
- logical: < > <= >= == != %in%

The data types

- Character
- numeric :

  _ logical (0 FALSE 1 TRUE)

  _ factor

  _ integer

  _ double

  _ etc...

The data objects

- vector
- matrix
- array
- list
- data frame

The functions

There are commands that transform or produce new elements from a R object or a part of a R object.

There are shaped as follow : **name_of_the_function(main arguments, options)**

sum(x,na.rm==TRUE) make the sum of the element of vector named x. The option na.rm==TRUE excludes from the sum the NA values.

```
x<-c(1:6,NA,c(5:9)-2)
x
```

```
##  [1]  1  2  3  4  5  6 NA  3  4  5  6  7
```

```
sum(x)
```

```
## [1] NA
```

```
sum(x,na.rm=TRUE)
```

```
## [1] 46
```

Find some help about a function

Use the help window or just put ? in front the function name.

```
?t.test()
```

```
## démarrage du serveur d'aide httpd ... fini
```

The syntax

R base has a stacked syntax.

```
paste(mean(x[x<5],na.rm=TRUE),'on average')
```

```
## [1] "2.83333333333333 on average"
```

The syntax

R base has a stacked syntax.

```
x
```

```
##  [1]  1  2  3  4  5  6 NA  3  4  5  6  7
```

```
x<5
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE    NA  TRUE  TRUE FALSE FALSE FALSE
```

```
x[x<5]
```

```
## [1]  1  2  3  4 NA  3  4
```

```
mean(x[x<5],na.rm=TRUE)
```

```
## [1] 2.833333
```

```
paste(mean(x[x<5],na.rm=TRUE),'on average')
```

```
## [1] "2.83333333333333 on average"
```

Intall and use packages

With R you have access to a wide panel of functions developped by the community. You can download packages of new functions from online repositories (like the CRAN, or github...).

Let's download the tidyverse package which includes a lot of usefull complementary fonctionality.

```
install.packages("tidyverse")
```

Intall and use packages

Once a package downloaded, you have to call it in your environment in order to use the included functions.

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✓ ggplot2 3.3.6      ✓ purrr   0.3.4
## ✓ tibble  3.1.8      ✓ dplyr   1.0.10
## ✓ tidyr   1.2.0      ✓ stringr 1.4.1
## ✓ readr   2.1.2      ✓ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

Intall and use packages

The tidyverse provides (among many other things) a usefull programing operator the simplify R syntax : the pipe operator %>% (keyboard shortcut 'ctrl' 'shift' 'm').

```
x<-1:5
y<-8:4
data.frame(x,y) %>% summarise(m_x=mean(x),m_y=mean(y))
```

```
##   m_x m_y
## 1   3   6
```

Intall and use packages

It is simpler than :

```
lapply(data.frame(x,y), mean)
```

```
## $x
## [1] 3
##
## $y
## [1] 6
```

```
sapply(data.frame(x,y), mean)
```

```
## x y
## 3 6
```

```
apply(data.frame(x,y),2, mean)
```

```
## x y
## 3 6
```

The home made functions

You can create you own function using the following syntax.

function_name<-function(arguements,...){

one program that do the job!!

}

The home made functions

Let's build a function that compute the actual value of a given amount of money in n compounding periods when the opportinity interest rate is i.

The home made functions

Let's do the math.

$$Vf = VA \times (1+i)^n$$

$$VA = \frac{VF}{(1+i)^n}$$

The home made functions

Let's do the program.

```
val_act<-function(vf,i,d){
  va<-vf*(1/(1+i)^d)
  return(va)
}
```

Let's test it.

```
val_act(1159.2741,0.03,5)
```

```
## [1] 1000
```

The home made functions

Create a function that computes the annual interest rate allowing to obtain the final value from the initial one in n periods. (5 minutes)

The home made functions

Let's do the math.

$$Vf = VA \times (1+i)^n$$

$$\frac{Vf}{VA} = (1+i)^n$$

$$\left(\frac{Vf}{VA}\right)^{1/n} = 1+i$$

$$i = \left(\frac{Vf}{VA}\right)^{1/n} - 1$$

The home made functions

Let's do the programing.

```
interet<-function(vf,va,n){
  i<-(vf/va)^(1/n)-1
  return(i)
}
```

Let's test it.

```
interet(1159.2741,1000,5)
```

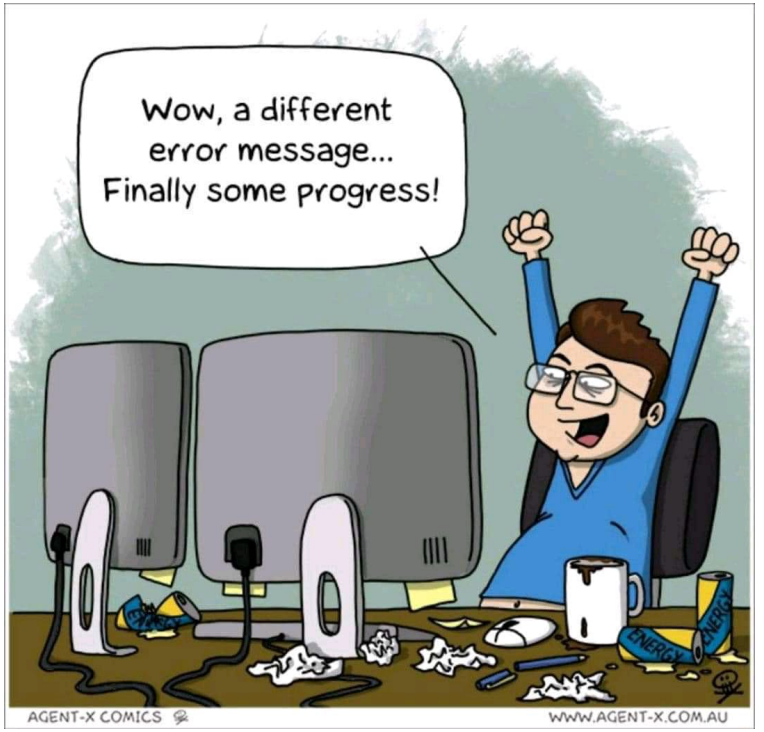
```
## [1] 0.03
```

More on R langage

You can watch (later) a 40 minutes video that presentes more extensively the points that we have just review. You can find it at the followning adress:

https://youtu.be/hGffaVR4Dtk

short break


Wow, a different error message... Finally some progress!
ENERGY
ENERGY
AGENT-X COMICS
WWW.AGENT-X.COM.AU

WRDS data base

Access to WRDS

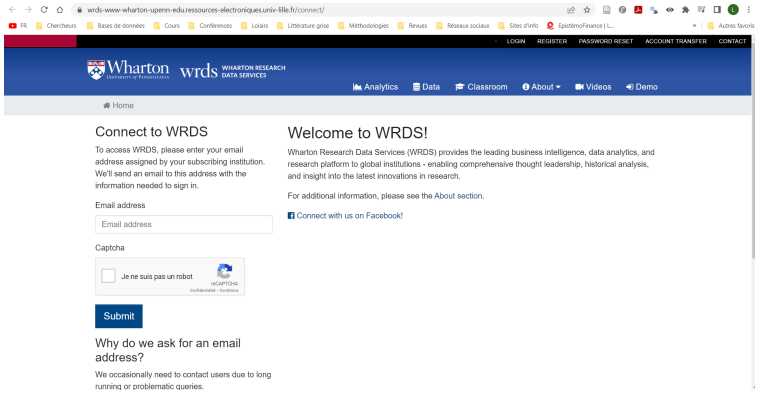
You start from your Student ENT. You select "toutes les applications" and on the part dedicated to 'Bibliothèque' you click on 'Ressources électronques'.

ent -> toutes les applications -> Bibliothèque -> ressources électroniques

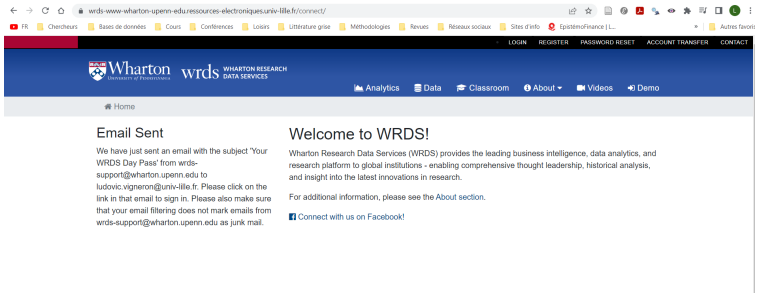
Once the 'ressources électroniques' page opens go to 'Accueil' and section the 'gestion' discipline. You can find the link with WRDS at the bottom of the current page.

Accueil -> gestion -> wrds

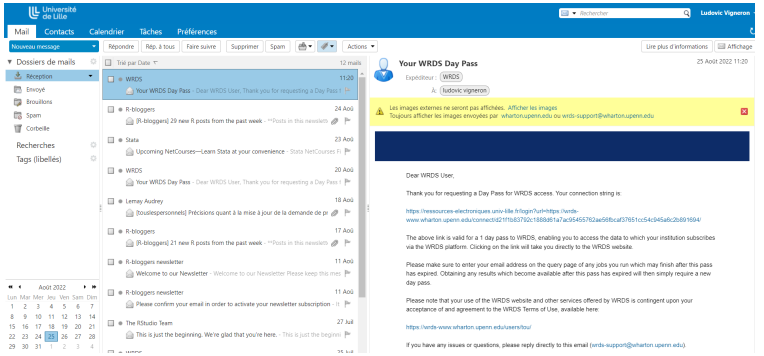
Access to WRDS



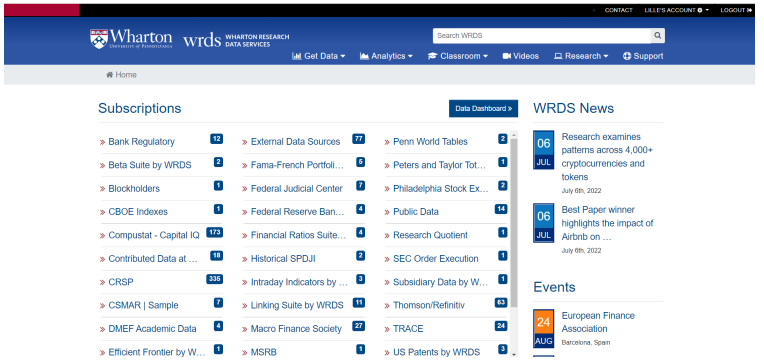
Access to WRDS



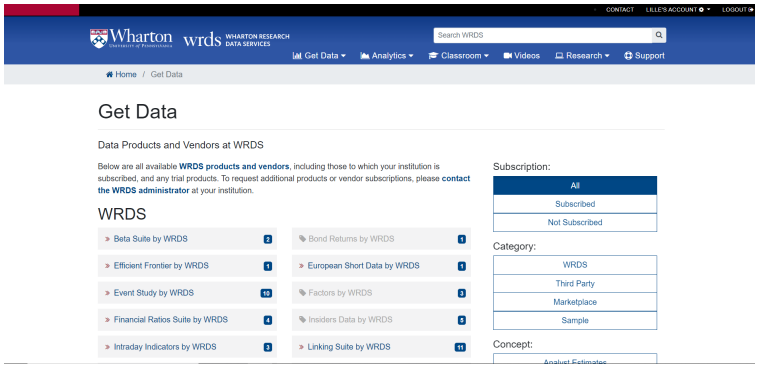
Access to WRDS



Access to WRDS



Access to WRDS



Access to WRDS

You only have access to ressources marked by the plain blue links.

If you spend to much time without doing anything, you will be disconnected.

Your access is can be activated using the email link during a day long.

What WRDS is?

It is a data base aggregator provided by the Wharton school of business of the university of Pennsylvania.

It allows to access multiple ressources in one place using a central data extraction interface.

It also provides some data treatement services (compute event studies, efficient portfolio, Beta...).

What the main ressources available are ?

- Compustat : accounting data about listed firms
- CRPS : share prices
- Thomson/Refinitiv (WRDS-Reuter Dealscan) : syndicated loan deals
- Trace : bonds

etc...

What the main ressources available are ?

Each database has it own indivuals' (company or security or CEO...) identifier.

One of the first difficulties that you will have to deal with using those data base will be to manage those identifiers in order to merge the data accuratly.

WRDS helps providing tables with merging keys (most of times).

Extracting/importing data

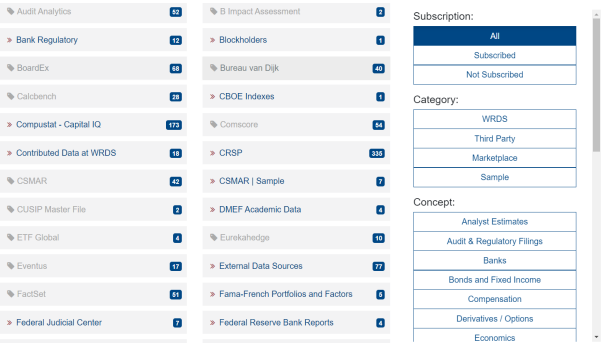
Extracting data from CRSP

The Center for Research in Security Prices, LLC (CRSP) maintains the most comprehensive collection of security price, return, and volume data for the NYSE, AMEX and NASDAQ stock markets.

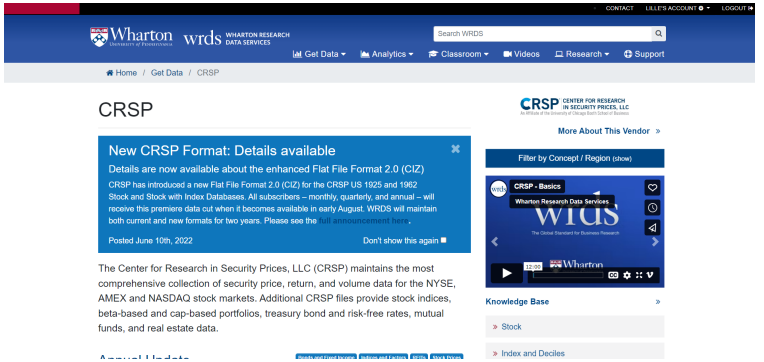
It is the gold standard data base on stocks in empirical finance (in US).

Most of the more important research papers published in top scientific journals (JF,JFE,JFQA...) have been produced using CRSP.

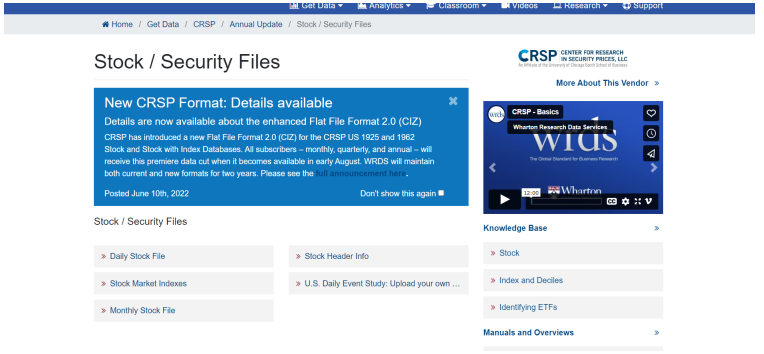
Extracting data from CRSP



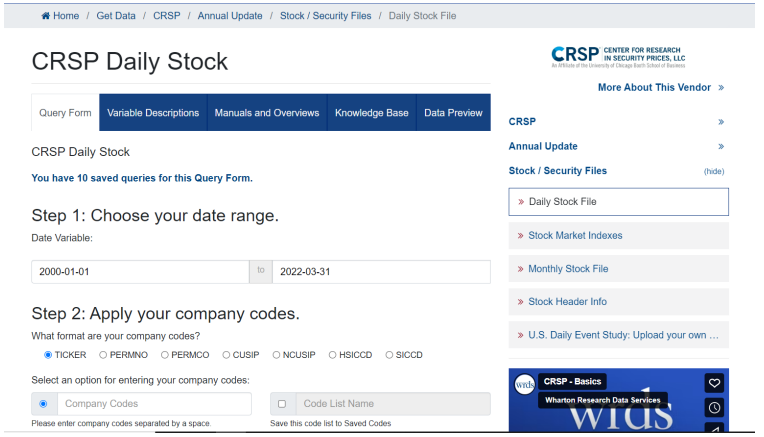
Extracting data from CRSP



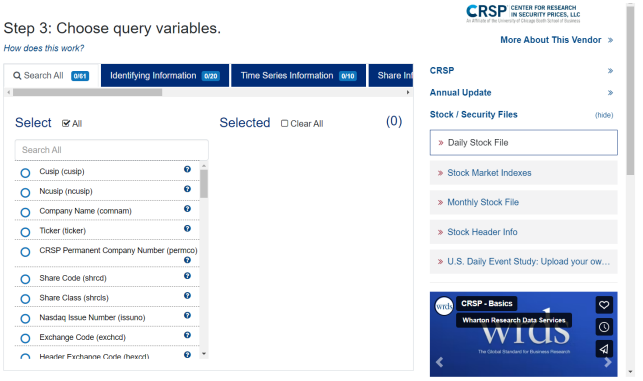
Extracting data from CRSP



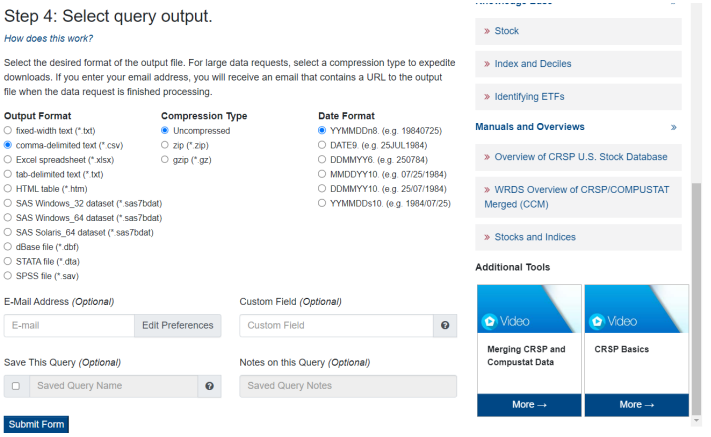
Extracting data from CRSP



Extracting data from CRSP



Extracting data from CRSP



Extracting data from CRSP

Mission 1:

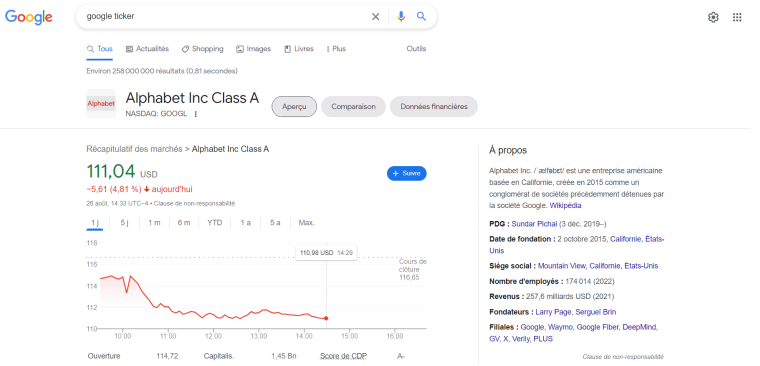
Find the ticker of a stock of your choice and put it in the chat window (company name followed by the corresponding actual ticker).

Extracting data from CRSP

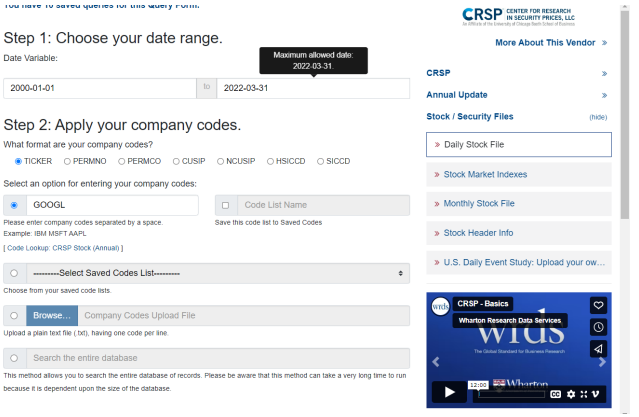
Let's take Google.

A quick research on it provides us its ticker.

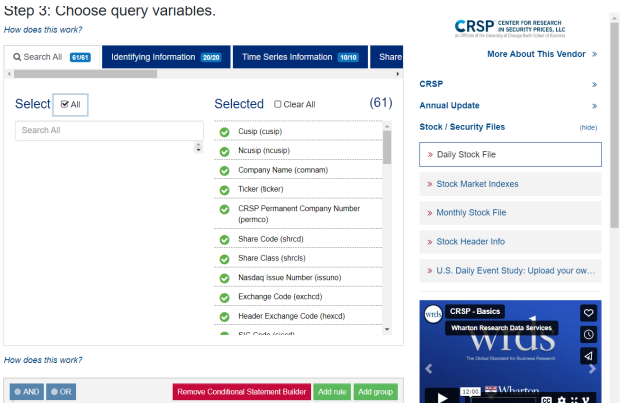
Extracting data from CRSP



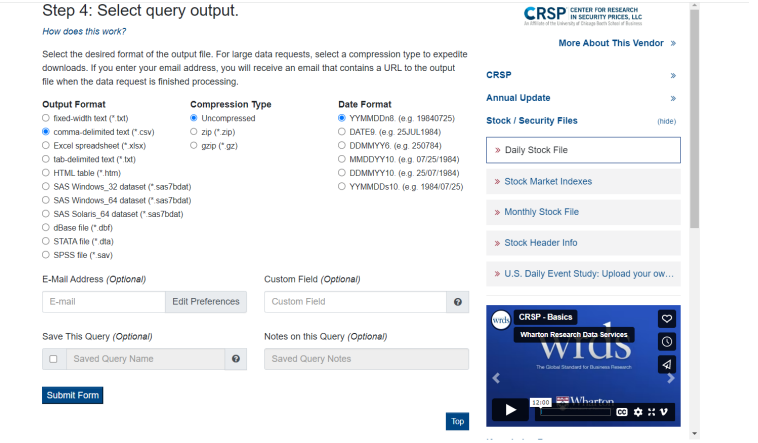
Extracting data from CRSP



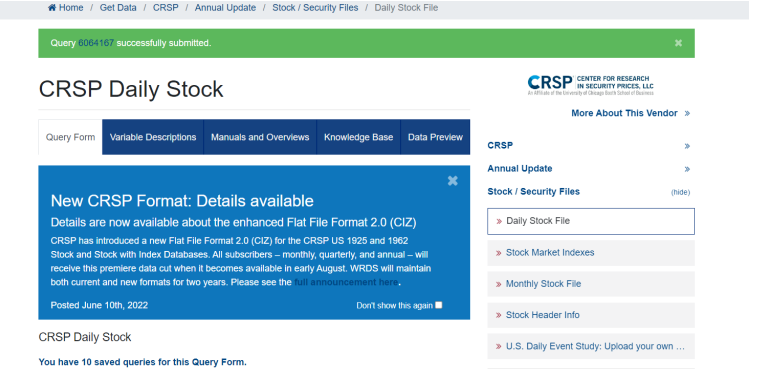
Extracting data from CRSP



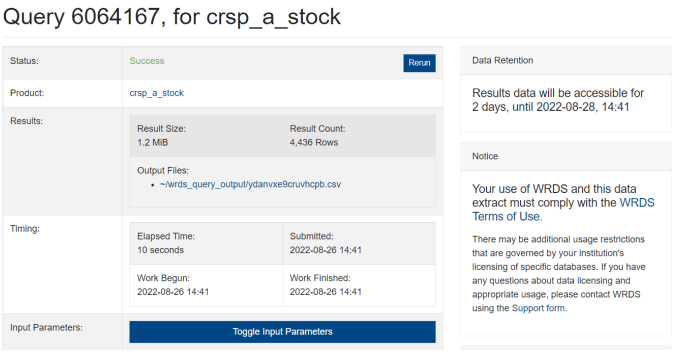
Extracting data from CRSP



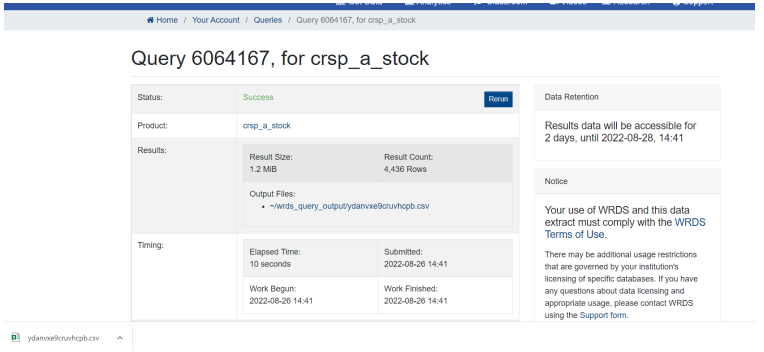
Extracting data from CRSP



Extracting data from CRSP



Extracting data from CRSP



Extracting data from CRSP

Copy the downloaded file in your working directory.

Once done import your data in R.

```
google <- read_csv("google.csv")
```

```
## Warning: One or more parsing issues, see `problems()` for details

## Rows: 4436 Columns: 63
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (13): NCUSIP, TICKER, COMNAM, SHRCLS, TSYMBOL, PRIMEXCH, TRDSTAT, SECSTA...
## dbl (39): PERMNO, date, NAMEENDT, SHRCD, EXCHCD, SICCD, NAICS, PERMCO, ISSUN...
## lgl (11): DCLRDT, DLPDT, NEXTDT, PAYDT, RCRDDT, DISTCD, DIVAMT, FACPR, FACSH...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Extracting data from CRSP

let's see what do we have. head(google)

Show 6 entries Search:

	PERMNO	date	NAMEENDT	SHRCD	EXCHCD	SICCD	NCUSIP	TICKER	COMN
1	90319	20040819	20050818	11	3	7375	38259P50	GOOG	GOOG INC
2	90319	20040820		11	3	7375	38259P50	GOOG	GOOG INC
3	90319	20040823		11	3	7375	38259P50	GOOG	GOOG INC
4	90319	20040824		11	3	7375	38259P50	GOOG	GOOG INC
5	90319	20040825		11	3	7375	38259P50	GOOG	GOOG INC
6	90319	20040826		11	3	7375	38259P50	GOOG	GOOG INC

Showing 1 to 6 of 6 entries Previous 1 Next

Extracting data from CRSP

```
unique(google$PERMNO)
```

```
## [1] 90319
```

```
unique(google$PERMCO)
```

```
## [1] 45483
```

```
unique(google$TICKER)
```

```
## [1] "GOOG"  "GOOGL"
```

```
unique(google$CUSIP)
```

```
## [1] "02079K30"
```

```
unique(google$COMNAM)
```

```
## [1] "GOOGLE INC"   "ALPHABET INC"
```

Extracting data from CRSP

PERMNO (sercurity) and PERMCO (company) are unique identifier given once by crsp respectivily to a security and a company they never change and are never reaffected.

There is no survivor biais in CRSP.

Extracting data from CRSP

Put the date variable on a date format.

```
str(google$date)
```

```
##  num [1:4436] 2e+07 2e+07 2e+07 2e+07 2e+07 ...
```

Extracting data from CRSP

You have to install the lubridate package to manage easily date format variables.

```
install.packages(lubridate)
```

Once the new package installed, call it in your working environment.

```
library(lubridate)
```

```
##
## Attachement du package : 'lubridate'

## Les objets suivants sont masqués depuis 'package:base':
##
##     date, intersect, setdiff, union
```

Extracting data from CRSP

So you can use ymd() function to format the date variable.

```
google$date<-ymd(google$date)
```

```
str(google$date)
```

```
##  Date[1:4436], format: "2004-08-19" "2004-08-20" "2004-08-23" "2004-08-24" "2004-08-25" ...
```

Extracting data from CRSP

let's see what do we have now. head(google)

Show 6 entries Search:

	PERMNO	date	NAMEENDT	SHRCD	EXCHCD	SICCD	NCUSIP	TICKER	COMNA
1	90319	2004-08-19	20050818	11	3	7375	38259P50	GOOG	GOOGLE INC
2	90319	2004-08-20		11	3	7375	38259P50	GOOG	GOOGLE INC
3	90319	2004-08-23		11	3	7375	38259P50	GOOG	GOOGLE INC
4	90319	2004-08-24		11	3	7375	38259P50	GOOG	GOOGLE INC
5	90319	2004-08-25		11	3	7375	38259P50	GOOG	GOOGLE INC
6	90319	2004-08-26		11	3	7375	38259P50	GOOG	GOOGLE INC

Showing 1 to 6 of 6 entries Previous 1 Next

Extracting data from CRSP

Let's check if we have duplicated observations.

```
google<-google %>% arrange(PERMNO,date) %>%
  group_by(PERMNO,date) %>%
  mutate(dup=n()) %>% ungroup()
table(google$dup)
```

```
##
##    1
## 4436
```

Extracting data from CRSP

There is no duplicated observations. We can drop the dup variable.

```
google <- google %>% select(-dup)
```

Extracting data from CRSP

There is too much information in the actual data base. Let reduce it to the ones we need.

```
google<-google %>% select(PERMNO,COMNAM,date,PRC,RET,RETX,DIVAMT,DISTCD,
                          DCLRDT,RCRDDT,PAYDT)
```

Extracting data from CRSP

When did google start to be listed?

```
min(google$date)
```

```
## [1] "2004-08-19"
```

Created in september 1998.

Did Google (Alphabet) pay some dividends since its listing?

```
unique(google$DIVAMT)
```

```
## [1] NA
```

Extracting data from CRSP

No dividence, we can also drop the related information.

```
google<-google %>% select(PERMNO,COMNAM,date,PRC,RET,RETX)
```

Extracting data from CRSP

Let's check if RET and RETX are always equal.

```
sum(google$RET==google$RETX)
```

```
## [1] 4436
```

We can drop RETX.

```
google<-google %>% select(-RETX)
```

Extracting data from CRSP

How are returns established in CRSP?

Simple return

$$r_t = \frac{P_1 - P_0}{P_0} = \frac{P_1}{P_0} - 1$$

```
google$PRC[2]/google$PRC[1]-1
```

```
## [1] 0.07948373
```

```
google$RET[2]
```

```
## [1] "0.079484"
```

CRPS provides simple retrurns.

Extracting data from CRSP

Returns (RET) are in character format. Lets transform the variable to make it numeric.

```
google$RET[1:10]
```

```
##  [1] "C"         "0.079484"  "0.010064"  "-0.041408" "0.010775"  "0.018019"
##  [7] "-0.016310" "-0.039001" "0.003529"  "-0.020709"
```

```
google$RET<-as.numeric(google$RET)
```

```
## Warning: NAs introduits lors de la conversion automatique
```

```
google$RET[1:10]
```

```
##  [1]         NA  0.079484  0.010064 -0.041408  0.010775  0.018019 -0.016310
##  [8] -0.039001  0.003529 -0.020709
```

Extracting data from CRSP

Let compute our own simple returns for the entier data base and compare with the precomputed returns

```
google <- google %>% arrange(date) %>%
  mutate(ret_s=round(PRC/lag(PRC)-1,digits=6),
         test=RET==ret_s)
google
```

```
## # A tibble: 4,436 × 7
##    PERMNO COMNAM     date         PRC      RET    ret_s test
##     <dbl> <chr>      <date>     <dbl>    <dbl>    <dbl> <lgl>
##  1  90319 GOOGLE INC 2004-08-19  100. NA       NA       NA
##  2  90319 GOOGLE INC 2004-08-20  108.  0.0795   0.0795  TRUE
##  3  90319 GOOGLE INC 2004-08-23  109.  0.0101   0.0101  TRUE
##  4  90319 GOOGLE INC 2004-08-24  105. -0.0414  -0.0414  TRUE
##  5  90319 GOOGLE INC 2004-08-25  106   0.0108   0.0108  TRUE
##  6  90319 GOOGLE INC 2004-08-26  108.  0.0180   0.0180  TRUE
##  7  90319 GOOGLE INC 2004-08-27  106. -0.0163  -0.0163  TRUE
##  8  90319 GOOGLE INC 2004-08-30  102. -0.0390  -0.0390  TRUE
##  9  90319 GOOGLE INC 2004-08-31  102.  0.00353  0.00353 TRUE
## 10  90319 GOOGLE INC 2004-09-01  100. -0.0207  -0.0207  TRUE
## # … with 4,426 more rows
```

Extracting data from CRSP

Let see if the two variables always match.

```
sum(google$test,na.rm=TRUE)
```

```
## [1] 4408
```

It is not the case.

```
google %>% filter(test==FALSE)
```

```
## # A tibble: 27 × 7
##    PERMNO COMNAM     date         PRC        RET      ret_s test
##     <dbl> <chr>      <date>     <dbl>      <dbl>      <dbl> <lgl>
##  1  90319 GOOGLE INC 2004-10-21  149.  0.0633     0.0633   FALSE
##  2  90319 GOOGLE INC 2005-01-11  194. -0.00779   -0.00779  FALSE
##  3  90319 GOOGLE INC 2005-02-01  192. -0.0190    -0.0190   FALSE
##  4  90319 GOOGLE INC 2005-02-16  198.  0.0163     0.0163   FALSE
##  5  90319 GOOGLE INC 2005-10-28  358.  0.0145     0.0145   FALSE
##  6  90319 GOOGLE INC 2006-12-04  485.  0.00842    0.00842  FALSE
##  7  90319 GOOGLE INC 2007-04-13  466. -0.00235   -0.00235  FALSE
##  8  90319 GOOGLE INC 2007-11-19  626. -0.0123    -0.0123   FALSE
##  9  90319 GOOGLE INC 2008-07-28  477. -0.0302    -0.0302   FALSE
## 10  90319 GOOGLE INC 2008-10-31  359. -0.000918 -0.000917 FALSE
## # … with 17 more rows
```

It is just marginal rounding problems.

Extracting data from CRSP

Let consider the continuous returns starting with some math.

$$vf = va.\, e^{r.n}$$

$$\frac{vf}{va} = e^{r.n}$$

$$ln\left(\frac{vf}{va}\right) = ln(e^{r.n})$$

$$r.n = ln\left(\frac{vf}{va}\right)$$

$$r = \frac{1}{n}.ln\left(\frac{vf}{va}\right)$$

here we just consider one period (so n=1).

$$r_t = \ln(\frac{P_1}{P_0})$$

Extracting data from CRSP

Let compute the continuous daily returns for Google.

```
google<-google %>% mutate(ret_c=log(PRC/lag(PRC)))
```

Extracting data from CRSP

Let compute the final value after one period using continuous returns.

```
google$PRC[1]*exp(google$ret_c[2])
```

```
## [1] 108.31
```

Extracting data from CRSP

Let consider now multiperiods returns.

Compute the return between date 1 **[2004-08-19]** and date 4 **[2004-08-24]**.

```
google$PRC[1]
```

```
## [1] 100.335
```

```
google$PRC[4]
```

```
## [1] 104.87
```

Extracting data from CRSP

Let consider now multiperiods returns.

simple return

```
google$PRC[4]/google$PRC[1]-1
```

```
## [1] 0.04519858
```

```
google$PRC[1]*(1+(google$PRC[4]/google$PRC[1]-1))
```

```
## [1] 104.87
```

continuous returns

```
log(google$PRC[4]/google$PRC[1])
```

```
## [1] 0.0442069
```

```
google$PRC[1]*exp(log(google$PRC[4]/google$PRC[1]))
```

```
## [1] 104.87
```

Extracting data from CRSP

simple returns intra period

```
google$PRC[1]*(1+google$ret_s[2])*(1+google$ret_s[3])*(1+google$ret_s[4])
```

```
## [1] 104.87
```

using cumprod()

```
google$PRC[1]*cumprod(1+google$ret_s[2:4])[3]
```

```
## [1] 104.87
```

Extracting data from CRSP

simple returns intra period

using geometric average

```
google$PRC[1]*(prod(1+google$ret_s[2:4])^(1/3))^3
```

```
## [1] 104.87
```

using a home made function for geometric average.

```
geomet_moy<-function(x){
  y<-prod(1+x)^(1/length(x))
  return(y)
}
```

```
google$PRC[1]*geomet_moy(google$ret_s[2:4])^3
```

```
## [1] 104.87
```

Extracting data from CRSP

continuous returns intra period

```
google$PRC[1]*exp(sum(google$ret_c[2:4]))
```

```
## [1] 104.87
```

```
google$PRC[1]*exp(mean(google$ret_c[2:4])*3)
```

```
## [1] 104.87
```

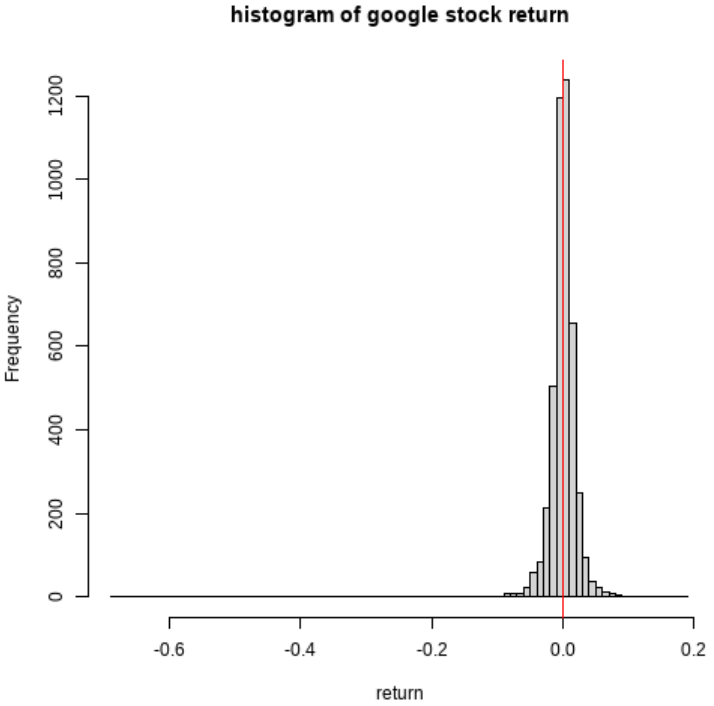
Extracting data from CRSP

Are Google return normaly distributed?

Extracting data from CRSP

Let see it drawing a histogram.

```
hist(google$ret_c,nclass = 100,
main="histogram of google stock return",xlab='return')
abline(v=mean(google$ret_c,na.rm=TRUE),col='red')
```


histogram of google stock return
Frequency
0
200
400
600
800
1000
1200
-0.6
-0.4
-0.2
0.0
0.2
return

Extracting data from CRSP

To be sure, let perform a Shapiro-Wilk test.

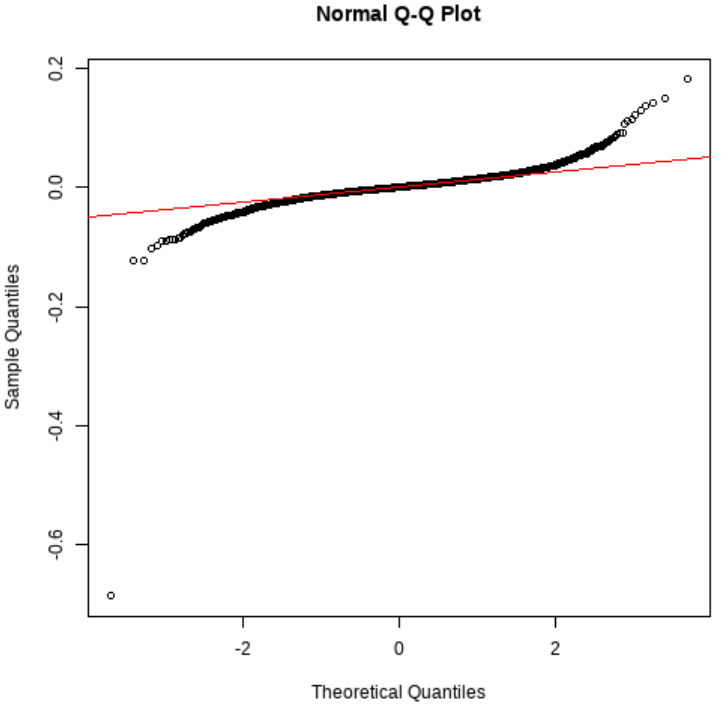
```
shapiro.test(google$ret_c)
```

```
##
##     Shapiro-Wilk normality test
##
## data:  google$ret_c
## W = 0.74424, p-value < 2.2e-16
```

Extracting data from CRSP

We also can examine the normality with a quantile-quantile plot (QQplot).

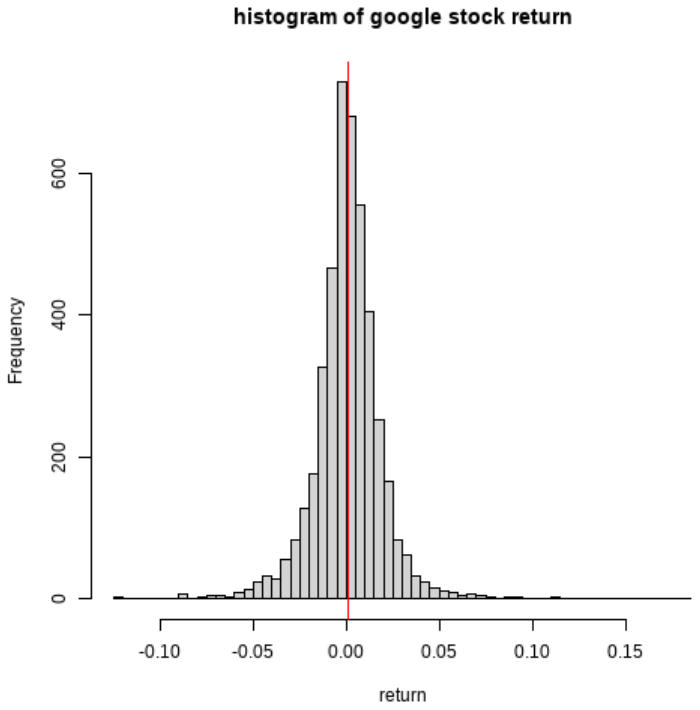
```
qqnorm(google$ret_c)
qqline(google$ret_c,col='red')
```


Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles
0.2
0.0
-0.2
-0.4
-0.6
-2
0
2

Extracting data from CRSP

Let drop the more extrem observation.

```
hist(google$ret_c[which(google$ret_c>-0.68)],nclass = 100,
     main="histogram of google stock return",xlab='return')
abline(v=mean(google$ret_c[which(google$ret_c>-0.68)],
              na.rm=TRUE),col='red')
```


histogram of google stock return
Frequency
0
200
400
600
-0.10
-0.05
0.00
0.05
0.10
0.15
return

Extracting data from CRSP

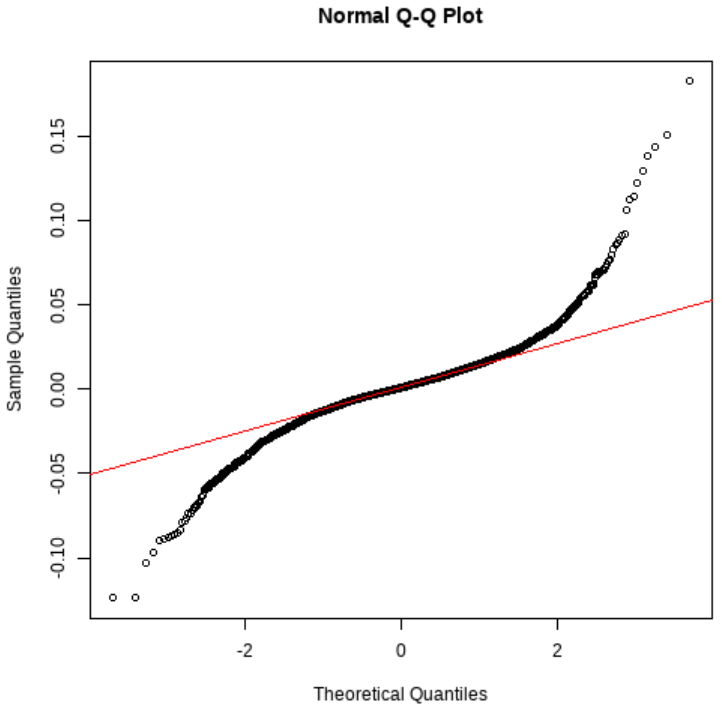
```
shapiro.test(google$ret_c[which(google$ret_c>-0.68)])
```

```
##
##     Shapiro-Wilk normality test
##
## data:  google$ret_c[which(google$ret_c > -0.68)]
## W = 0.90833, p-value < 2.2e-16
```

Extracting data from CRSP

```
qqnorm(google$ret_c[which(google$ret_c>-0.68)])
qqline(google$ret_c[which(google$ret_c>-0.68)],col='red')
```

Extracting data from CRSP



Extracting data from CRSP

Let estimate the CAPM Beta of Google.

What do we need?

- stock return
- market return
- risk free rate

$$E(r_{it}) = \alpha_i + \beta_i(rm_t - rf_t) + \epsilon_{i,t}$$

Extracting data from CRSP (market index)

Extracting data from CRSP (market index)

```
index_CRPS <- read_csv("index_CRPS.csv")
```

```
## Rows: 4594 Columns: 11
## ── Column specification ────────────────────────────────────────────────────
## Delimiter: ","
## dbl (11): DATE, vwretd, vwretx, ewretd, ewretx, sprtrn, spindx, totval, totc...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Extracting data from CRSP (market index)

Import data in R

```
index_CRPS$DATE<-ymd(index_CRPS$DATE)
```

Extracting data from CRSP (market index)

format data.

```
index_CRPS_<-index_CRPS %>% select(DATE,vwretd) %>%
  filter(DATE>="2004-08-19") %>%
  rename(date=DATE)
```

Extracting data from CRSP (market index)

merge the data with google returns.

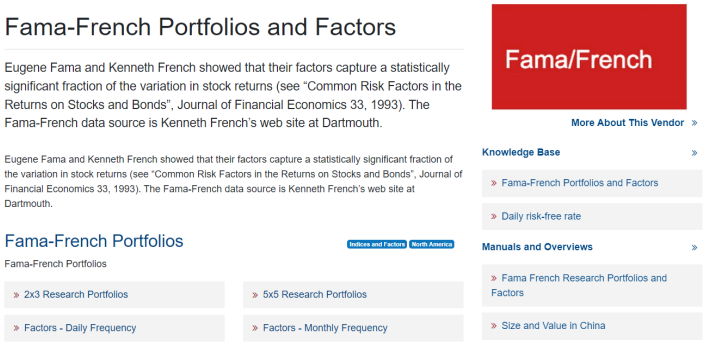
```
mer_base<-left_join(google,index_CRPS_)
```

```
## Joining, by = "date"
```

drop omit values (NA)

```
mer_base<-mer_base %>% na.omit()
```

Extracting data from Fama-French Portofolio and factors (risk free rate)



Fama-French Portfolios and Factors

Eugene Fama and Kenneth French showed that their factors capture a statistically significant fraction of the variation in stock returns (see "Common Risk Factors in the Returns on Stocks and Bonds", Journal of Financial Economics 33, 1993). The Fama-French data source is Kenneth French's web site at Dartmouth.

Eugene Fama and Kenneth French showed that their factors capture a statistically significant fraction of the variation in stock returns (see "Common Risk Factors in the Returns on Stocks and Bonds", Journal of Financial Economics 33, 1993). The Fama-French data source is Kenneth French's web site at Dartmouth.

Fama-French Portfolios

Indices and Factors North America

Fama-French Portfolios

» 2x3 Research Portfolios

» 5x5 Research Portfolios

» Factors - Daily Frequency

» Factors - Monthly Frequency

Fama/French

More About This Vendor »

Knowledge Base »

» Fama-French Portfolios and Factors

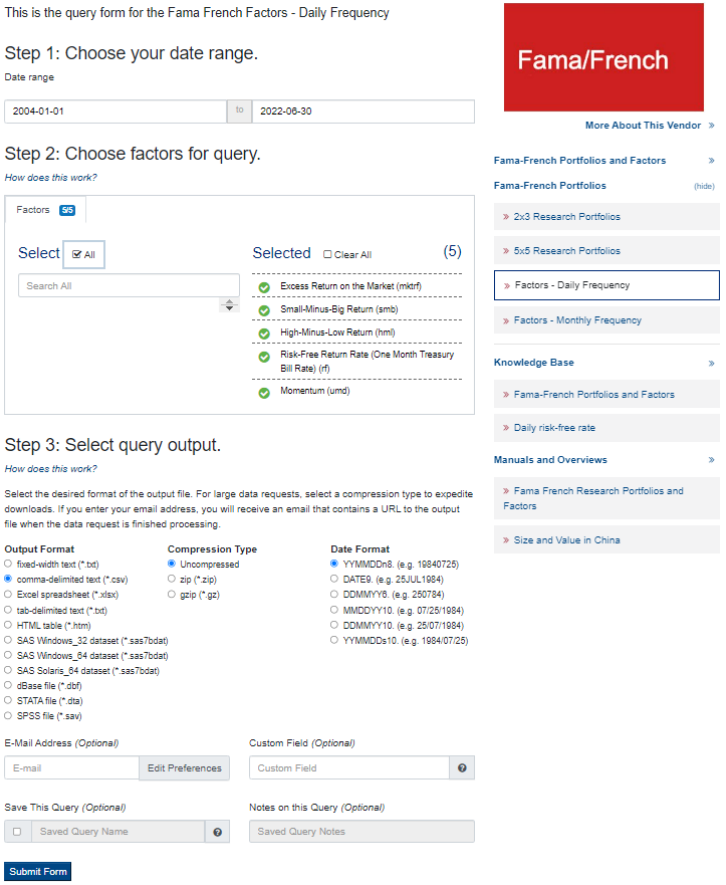
» Daily risk-free rate

Manuals and Overviews »

» Fama French Research Portfolios and Factors

» Size and Value in China

Extracting data from Fama-French Portofolio and factors (risk free rate)



Extracting data from Fama-French Portofolio and factors (risk free rate)

Now, we can import the new data in R

```
risk_free_rate <- read_csv("risk_free_rate.csv")
```

```
## Rows: 4656 Columns: 6
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (6): date, mktrf, smb, hml, rf, umd
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Extracting data from Fama-French Portofolio and factors (risk free rate)

and prepare them to be merged with our estimation sample data.

```
risk_free_rate <- risk_free_rate %>% select(date,rf) %>%
  mutate(date=ymd(date)) %>%
  filter(date>="2004-08-19"&date<='2022-03-31')
```

Extracting data from Fama-French Portofolio and factors (risk free rate)

Merge the data

```
mer_base<-left_join(mer_base,risk_free_rate)
```

```
## Joining, by = "date"
```

Estimating MEDAF BETA

Variables computation

- risk premium
- market risk premium

Estimating MEDAF BETA

```
mer_base<-mer_base %>% mutate(risp=RET-rf,
                              markrisp=vwretd-rf)
```

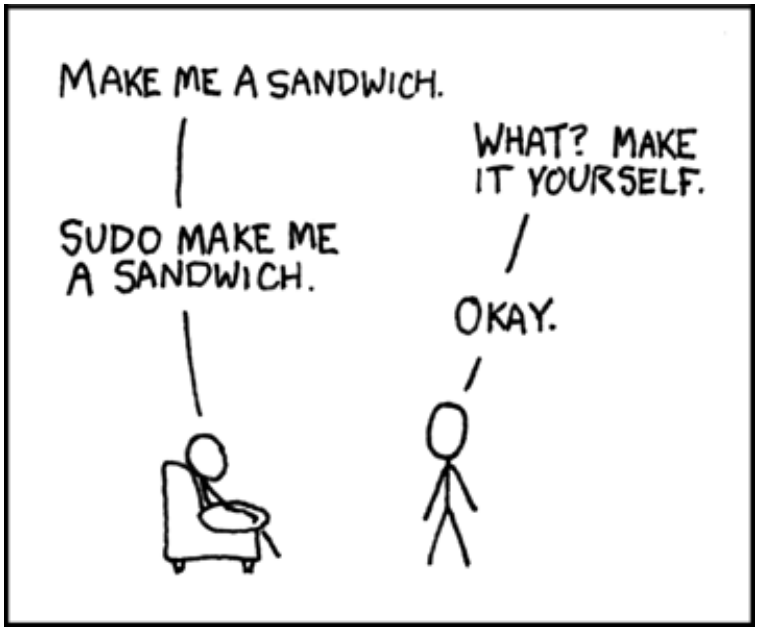
Estimating MEDAF BETA

Let limit us to a estimation over 130 cotation days.

```
rega<-lm(risp~markrisp,data=filter(mer_base,date<=date[130]))
summary(rega)
```

```
##
## Call:
## lm(formula = risp ~ markrisp, data = filter(mer_base, date <=
##     date[130]))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.090092 -0.021006  0.000732  0.013994  0.154748
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.004790   0.002821   1.698   0.0919 .
## markrisp    0.567737   0.451241   1.258   0.2106
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03182 on 128 degrees of freedom
## Multiple R-squared:  0.01222,    Adjusted R-squared:  0.004499
## F-statistic: 1.583 on 1 and 128 DF,  p-value: 0.2106
```

If you want more about importing and managing data with R, you can watch the following video https://youtu.be/1CSE0mQVwAQ (31 minutes).


MAKE ME A SANDWICH.
SUDO MAKE ME A SANDWICH.
WHAT? MAKE IT YOURSELF.
OKAY.