Financial Risk Forecasting
Seminar Week 4: Creating Reports with Quarto

Jon Danielsson
London School of Economics

Version 4.0

4 Making reports

4.1 Why reproducible reporting matters in finance

In financial analysis, creating reproducible reports is important for several reasons. Regulatory requirements often demand that analysis can be replicated and verified by third parties. Investment committees need consistent formatting and methodology when reviewing risk assessments. Client presentations require professional output that can be updated automatically as new data becomes available.

With Quarto, we embed R analysis directly into reports, ensuring that calculations, plots and statistics are always consistent with the underlying data. This reduces errors from manual copying and makes it easier to update reports when new data arrives.

Markdown-based reporting systems like Quarto are also becoming important in automated systems and AI workflows. Many AI tools can generate and process markdown documents, making it easier to integrate human analysis with automated reporting systems. This standardized format allows reports to be generated programmatically while maintaining professional presentation standards.

4.2 Loading data and libraries

```
# Install packages if needed
# install.packages(c("tseries", "car", "lubridate", "moments", "knitr"), repos = "https://

library(tseries)
library(car)
library(lubridate)
library(zoo)
library(moments)
```

```
library(knitr)
```

```
load('Returns.RData')
load('Prices.RData')
load('NormalisedPrices.RData')
load('security_names.RData')
load('tickers.RData')
```

4.3 Links from the R notebook

Presentations and reports from the R notebook.

4.4 Word and PowerPoint vs. Quarto

Most people use Microsoft Word or PowerPoint to do reports. They create the document in that software and then copy the output from R into the document. Usually that is the best way, but there is an alternative that can be useful. The company that provides RStudio has a product called Quarto. One advantage is that this is often how AI systems make reports. We can automatically use Quarto to make Word, PowerPoint, html and PDF files with RStudio.

4.5 Example Quarto project

We have created an example project that runs a small set of analysis. It is kept on github.com/Jon-Danielsson/Financial-Risk-Forecasting-Example-Project.

4.6 To start

Make a Quarto file and save it. In RStudio, File -> New file -> Quarto Document or Quarto Presentation. Save file to same place your data is in.

At the top of the file you find setup instructions. `format` indicates how the output ends up.

When making reports, it is usually best to stick with webpages, i.e. `html` while developing the report, only to change to word/PowerPoint/pdf when making final report.

```
---
title: "Seminar 4. Documents"
format: html
editor: visual
---
```

But when making final reports, switch to pdf:

```
format: pdf
```

If you make pdf reports, you need some extra packages, since PDF output requires a LaTex installation. LSE computers have it installed, but if you need it, the easiest is yihui.org/tinytex.

Or Word:

```
format: docx
```

For pdf slides:

```
format: beamer
```

And PowerPoint:

```
format: pptx
```

And finally, interactive slides:

```
format: revealjs
```

```
install.packages('tinytex', repos = "https://cloud.r-project.org/")
tinytex::install_tinytex()
```

4.7 Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

If your quarto document contains

````
```{r}
a=34
b=0.4
c=a^b
print(c)
```
````

You get:

```
a=34
b=0.4
c=a^b
cat(c)
```

```
4.098185
```

Then you can put these results inside text. Suppose you write:

```
If we start with `r a`  and raise
it to the power of `r b`, we get `r c`,
rounded as `r round(c,2)`.
```

It comes out as:

If we start with 34 and raise it to the power of 0.4, we get 4.0981851 , rounded as 4.1.

4.8 Working with data frames

First, we can see what data we have. If your quarto document contains:

```{r}
names(Returns)
```

You get:

```
names(Returns)
```

```
 [1] "date" "GSPC" "IXIC" "AAPL" "MSFT" "JPM"  "C"    "XOM"  "MCD"  "GE"
[11] "NVDA"
```

Now we can create more sophisticated output. Suppose you write:

```{r}
#| echo: false
tickers_temp=names(Returns)
tickers_temp=tickers_temp[2:length(tickers_temp)]
cat("The firms we have are:")
for(i in tickers_temp) cat(i,", ")
cat("\n")
```

The `#| echo: false` hides the code and only shows the output:

```
The firms we have are:
```

```
GSPC , IXIC , AAPL , MSFT , JPM , C , XOM , MCD , GE , NVDA ,
```

4.9 Basic dynamic text with calculations

We can make our reports dynamic by embedding calculations directly in the text. First, select a stock to analyse:

```{r}
#| echo: false
use="AAPL"
```

4.9.1 Simple inline statistics

Now we can write text that automatically updates with our data. If you write:

```
If we take `r use`, we have `r length(Returns[[use]])`
observations. The mean return is
`r mean(Returns[[use]])*100`%, and
on the best day the return was
`r round(max(Returns[[use]])*100,1)`%.
```

It comes out as:

If we take AAPL, we have 6439 observations. The mean return is 0.0884201%, and on the best day the return was 14.3%.

4.9.2 Finding specific dates

We can also find specific dates. If you write:

```
That happened on
`r Returns$date[Returns[[use]]==max(Returns[[use]])]`.
Can format that as
`r format(ymd(Returns$date[Returns[[use]]==max(Returns[[use]])]), "%d %B %Y")`
```

You get:

That happened on 2025-04-09. Can format that as 09 April 2025

Similarly for the worst day:

```
The worst day in `r use`'s history happened on
`r format(ymd(Returns$date[Returns[[use]]==min(Returns[[use]])]), "%d %B %Y")`
when it fell by `r round(min(Returns[[use]])*100,1)`%.
```

Produces:

The worst day in AAPL's history happened on 29 September 2000 when it fell by -73.1%.

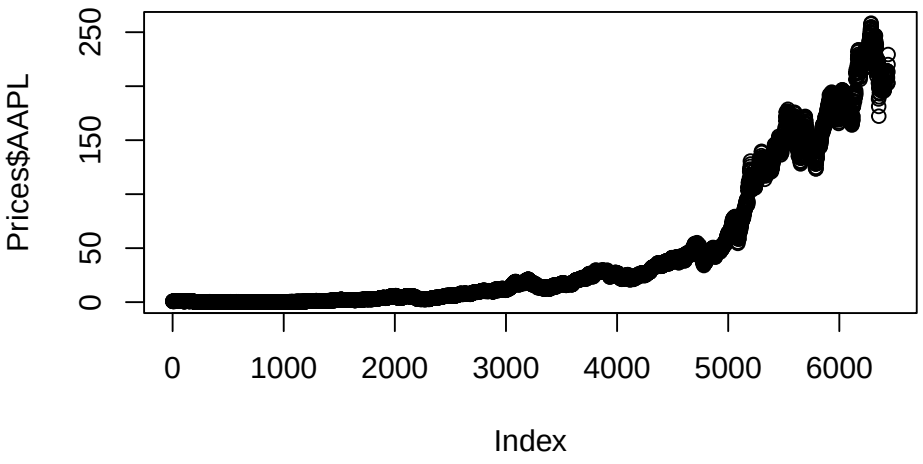
4.10 Plotting Apple

Basic plots are straightforward. This code:

```
```{r}
plot(Prices$AAPL)
```
```

Produces:

```
plot(Prices$AAPL)
```



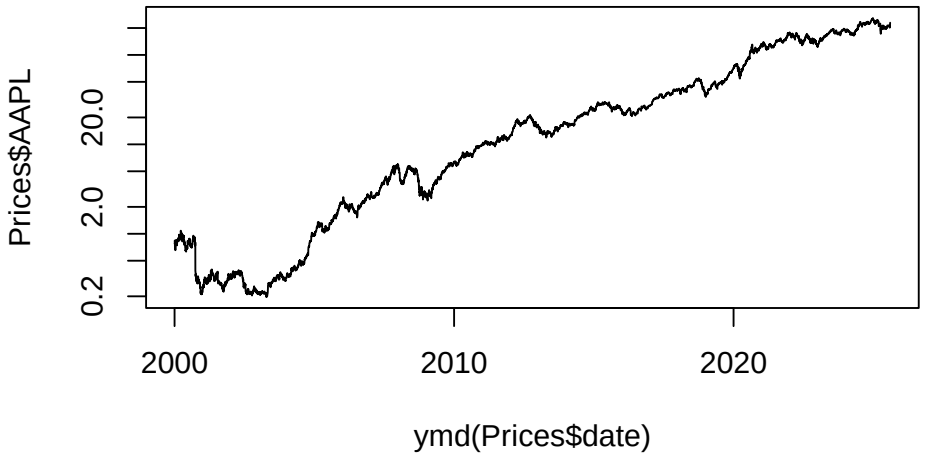
4.11 Plotting Apple with dates

We can improve plots by formatting dates properly. If you write:

```{r}
plot(ymd(Prices$date),Prices$AAPL,type='l',log='y')
```

This produces a much nicer plot with dates on the x-axis and log scale on y-axis:

```
plot(ymd(Prices$date),Prices$AAPL,type='l',log='y')
```



4.12 Working with multiple assets

4.12.1 Basic text output for all assets

We can create summary statistics programmatically. This code:

```{r}
#| echo: false
for(i in tickers){
  x=Returns[[i]]
  cat(i,mean(x),sd(x),min(x),max(x),"\n")
}
```

Produces a text output (with code hidden):

```
GSPC 0.0002282802 0.01227778 -0.1276522 0.109572
IXIC 0.0002581538 0.01576297 -0.1314916 0.1325464
AAPL 0.0008842013 0.02524584 -0.731433 0.1426175
MSFT 0.0004163974 0.01898984 -0.169579 0.1786917
JPM 0.0003821615 0.02330617 -0.2322796 0.2239187
C -0.0001338934 0.02913396 -0.4946965 0.4563144
XOM 0.000271038 0.01657936 -0.1502706 0.1586315
MCD 0.0004110747 0.01437685 -0.1728701 0.1665768
GE 0.0001114734 0.02092539 -0.1643972 0.1798418
NVDA 0.001183278 0.03727013 -0.4341752 0.3537162
```

4.12.2 Creating formatted tables

For better formatting, we can create a proper table. This more complex code:

```
```{r}
#| echo: false
df=matrix(ncol=4,nrow=length(n))
n=names(Returns)
n=n[2:length(n)]

for(i in 1:length(n)){
  x=Returns[[n[i]]]
  df[i,]=c(mean(x),sd(x),min(x),max(x))
}

df=as.data.frame(df)
df=cbind(n,df*100)
names(df)=c("Asset","mean","sd","min","max")

kable(df,digits=3,caption="Sample stats (in %)")
```
```

Creates a nicely formatted table using `kable()`:

Table 1: Sample stats (in %)

Asset	mean	sd	min	max
GSPC	0.023	1.228	-12.765	10.957
IXIC	0.026	1.576	-13.149	13.255
AAPL	0.088	2.525	-73.143	14.262
MSFT	0.042	1.899	-16.958	17.869
JPM	0.038	2.331	-23.228	22.392
C	-0.013	2.913	-49.470	45.631
XOM	0.027	1.658	-15.027	15.863
MCD	0.041	1.438	-17.287	16.658
GE	0.011	2.093	-16.440	17.984
NVDA	0.118	3.727	-43.418	35.372

4.13 Testing

We can perform statistical tests on our data. This code:

```
```{r}
y=Returns$GE
mean(y)
sd(y)
skewness(y)
kurtosis(y)
jarque.bera.test(y)
Box.test(y, type = "Ljung-Box")
```
```

```
Box.test(y^2, type = "Ljung-Box")
```

Runs various statistical tests on GE returns:

```
y=Returns$GE
mean(y)
```

```
[1] 0.0001114734
```

```
sd(y)
```

```
[1] 0.02092539
```

```
skewness(y)
```

```
[1] -0.06217339
```

```
kurtosis(y)
```

```
[1] 10.37314
```

```
jarque.bera.test(y)
```

```
    Jarque Bera Test

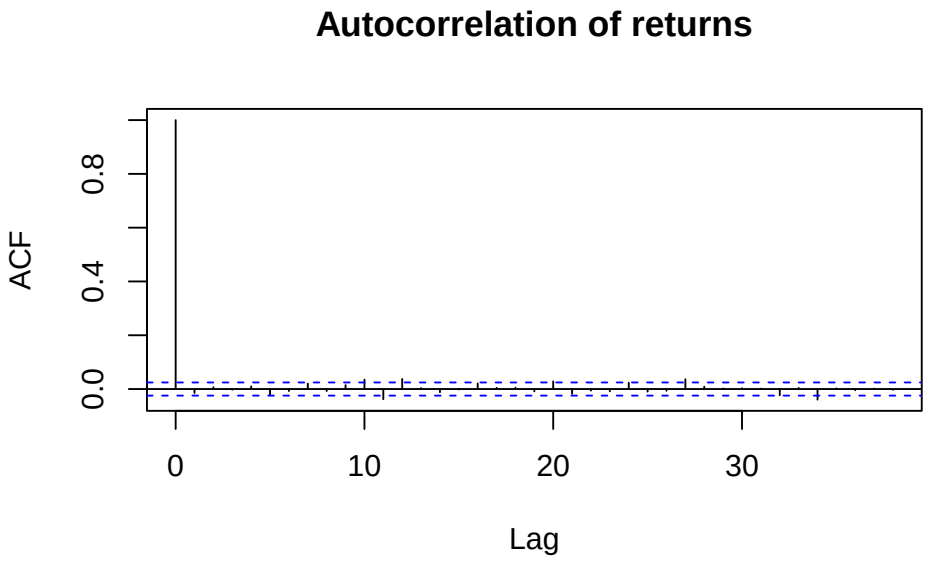
data:  y
X-squared = 14589, df = 2, p-value < 2.2e-16
```

```
Box.test(y, type = "Ljung-Box")
```

```
    Box-Ljung test

data:  y
X-squared = 1.4112, df = 1, p-value = 0.2349
```

```
Box.test(y^2, type = "Ljung-Box")
```

```
    Box-Ljung test

data:  y^2
X-squared = 432.81, df = 1, p-value < 2.2e-16
```

4.14 ACF

To visualize autocorrelation, use:

```
```{r}
acf(y, main = "Autocorrelation of returns")
```
```

Which produces:

```r
acf(y, main = "Autocorrelation of returns")
```

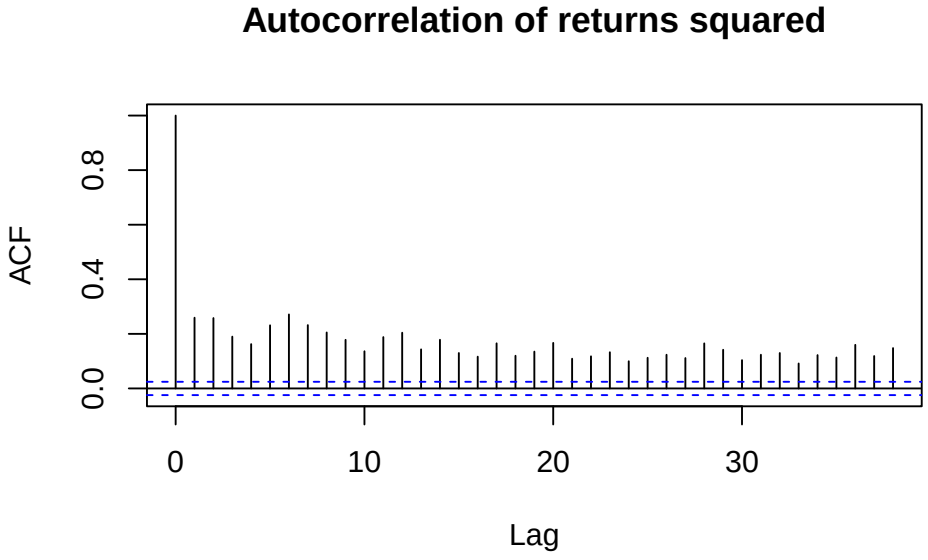


For squared returns (to check for volatility clustering):

```
```{r}
acf(y^2, main = "Autocorrelation of returns squared")
```
```

Gives:

```r
acf(y^2, main = "Autocorrelation of returns squared")
```



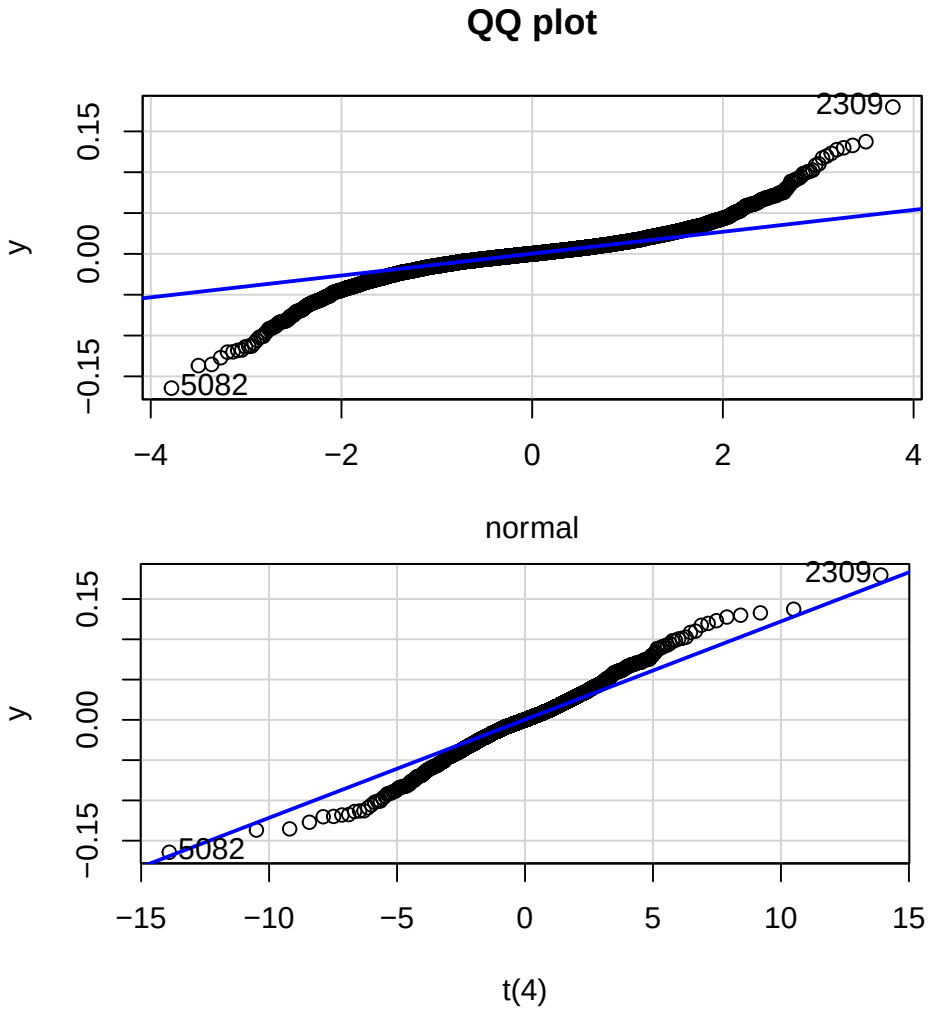
These ACF plots look like they could be nicer. The notebook shows some better alternatives.

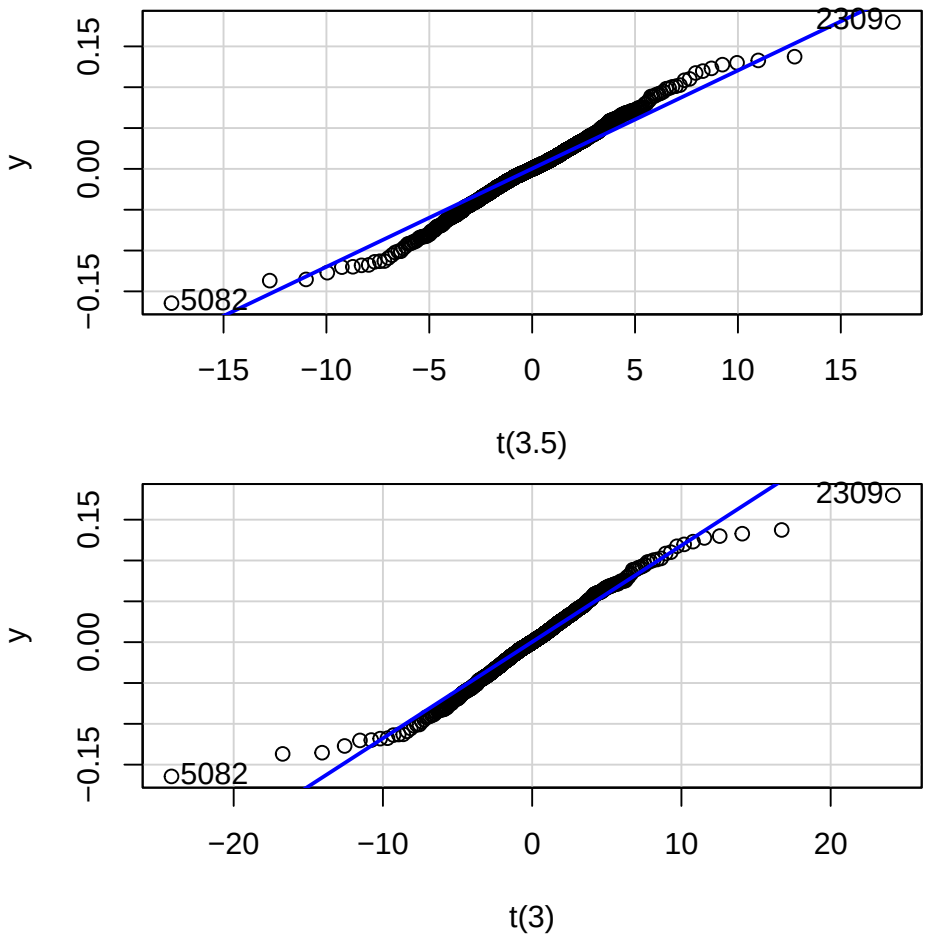
4.15 QQ Plots

Finally, we can create multiple QQ plots to assess distributional fit. This code (with output only):

```
```{r}
#| echo: false
x=qqPlot(y, distribution = "norm", envelope = FALSE,xlab="normal",main="QQ plot")
x=qqPlot(y, distribution = "t", df = 4, envelope = FALSE,xlab="t(4)")
x=qqPlot(y, distribution = "t", df = 3.5, envelope = FALSE,xlab="t(3.5)")
x=qqPlot(y, distribution = "t", df = 3, envelope = FALSE,xlab="t(3)")
```
```

Creates multiple QQ plots comparing the returns to different distributions:





4.16 Recap

4.16.1 In this seminar we have covered:

- Creating reports and presentations with Quarto
- Different output formats:
  - HTML for web-based reports
  - PDF for professional documents
  - Word (docx) for editable documents
  - PowerPoint (pptx) and Beamer for presentations
  - RevealJS for interactive slides
- Embedding R code in documents:
  - Code chunks with `{r}`
  - Inline R code with backtick-r syntax
  - Controlling output with `#| echo: false`
- Creating dynamic reports:
  - Embedding calculations in text
  - Automatically updating statistics
  - Generating tables with `kable()`
- Integrating statistical analysis:

- Summary statistics tables
- Statistical tests output
- ACF plots
- QQ plots for distribution assessment

4.16.2 Some new functions used:

- `kable()` — create formatted tables for reports
- `suppressPackageStartupMessages()` — hide package loading messages
- `knitr` library — powers the document rendering
- `for()` — loop through elements
- `round()` — round numbers to specified digits
- `matrix()` — create a matrix
- `as.data.frame()` — convert to data frame
- `cbind()` — combine columns
- `ymd()` — parse dates in year-month-day format
- `format()` — format dates and other objects for display

4.17 Optional exercises

1. **Basic Quarto document**:
   - Create a one-page HTML document analyzing the statistical properties of a single stock
   - Include: summary statistics, histogram of returns, ACF plot and QQ plot
   - Add inline text describing the highest and lowest returns with dates
   - Export the same document as PDF and Word formats
2. **Comparative analysis report**:
   - Create a report comparing two stocks from different sectors
   - Use a loop to generate summary statistics for both
   - Include side-by-side plots using `par(mfrow=c(1,2))`
   - Add a conclusion section using inline R to state which stock is more volatile
3. **Interactive presentation**:
   - Make a RevealJS presentation on market volatility
   - Each slide should focus on a different stock
   - Include one slide with animated transitions between plots
   - Convert the same content to PowerPoint and Beamer formats
4. **Automated reporting**:
   - Create a function that takes a ticker symbol and generates a complete report
   - The report should include all statistics from the seminar
   - Use conditional formatting: highlight returns $> 5\%$ in green, $< -5\%$ in red
   - Test with at least three different stocks
5. **Crisis analysis document**:
   - Create a document focusing on the 2008 financial crisis period
   - Use date filtering to show only 2007-2009 data
   - Create a table showing maximum drawdown for each stock
   - Include a timeline plot with `abline()` marking key crisis dates

6. **Professional report template**:
    - Design a reusable Quarto template for financial analysis
    - Include a YAML header with parameters for stock selection and date range
    - Add a table of contents and numbered sections
    - Create both light and dark theme versions
7. **Dynamic dashboard**:
    - Create an HTML document with multiple tabs (using Quarto's tabset feature)
    - Each tab analyzes a different aspect: prices, returns, volatility, correlations
    - Include a summary tab that updates automatically based on the selected stock
    - Add a date range selector that filters all plots
8. **Batch reporting**:
    - Write a script that generates individual PDF reports for each stock in your dataset
    - Each report should be saved as "StockAnalysis_TICKER_DATE.pdf"
    - Create an index HTML page that links to all generated reports
    - Include execution time tracking to optimise performance
9. **Advanced dashboard**:
    - Use `Shiny` to make an interactive dashboard to display prices, returns and analysis