15 Year stock data

Aditya Shah

2025-05-04

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.4.2

## Warning: package 'ggplot2' was built under R version 4.4.3

## Warning: package 'lubridate' was built under R version 4.4.2

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.5.2 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.3 ✔ tidyr 1.3.1  
## ✔ purrr 1.0.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(lubridate)  
library(xts)

## Loading required package: zoo  
##   
## Attaching package: 'zoo'  
##   
## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric  
##   
##   
## ######################### Warning from 'xts' package ##########################  
## # #  
## # The dplyr lag() function breaks how base R's lag() function is supposed to #  
## # work, which breaks lag(my\_xts). Calls to lag(my\_xts) that you type or #  
## # source() into this session won't work correctly. #  
## # #  
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #  
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop #  
## # dplyr from breaking base R's lag() function. #  
## # #  
## # Code in packages is not affected. It's protected by R's namespace mechanism #  
## # Set `options(xts.warn\_dplyr\_breaks\_lag = FALSE)` to suppress this warning. #  
## # #  
## ###############################################################################  
##   
## Attaching package: 'xts'  
##   
## The following objects are masked from 'package:dplyr':  
##   
## first, last

library(zoo)  
library(scales)

##   
## Attaching package: 'scales'  
##   
## The following object is masked from 'package:purrr':  
##   
## discard  
##   
## The following object is masked from 'package:readr':  
##   
## col\_factor

library(ggfortify)

## Warning: package 'ggfortify' was built under R version 4.4.3

library(tidyquant)

## Warning: package 'tidyquant' was built under R version 4.4.3

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

## Warning: package 'PerformanceAnalytics' was built under R version 4.4.3

## ── Attaching core tidyquant packages ─────────────────────── tidyquant 1.0.11 ──  
## ✔ PerformanceAnalytics 2.0.8 ✔ TTR 0.24.4  
## ✔ quantmod 0.4.26   
## ── Conflicts ────────────────────────────────────────── tidyquant\_conflicts() ──  
## ✖ zoo::as.Date() masks base::as.Date()  
## ✖ zoo::as.Date.numeric() masks base::as.Date.numeric()  
## ✖ scales::col\_factor() masks readr::col\_factor()  
## ✖ scales::discard() masks purrr::discard()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ xts::first() masks dplyr::first()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ xts::last() masks dplyr::last()  
## ✖ PerformanceAnalytics::legend() masks graphics::legend()  
## ✖ quantmod::summary() masks base::summary()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(PerformanceAnalytics)  
library(TTR)  
library(forecast)

## Warning: package 'forecast' was built under R version 4.4.2

## Registered S3 methods overwritten by 'forecast':  
## method from   
## autoplot.Arima ggfortify  
## autoplot.acf ggfortify  
## autoplot.ar ggfortify  
## autoplot.bats ggfortify  
## autoplot.decomposed.ts ggfortify  
## autoplot.ets ggfortify  
## autoplot.forecast ggfortify  
## autoplot.stl ggfortify  
## autoplot.ts ggfortify  
## fitted.ar ggfortify  
## fortify.ts ggfortify  
## residuals.ar ggfortify

library(tseries)  
library(timetk)

## Warning: package 'timetk' was built under R version 4.4.3

## Registered S3 method overwritten by 'parsnip':  
## method from   
## autoplot.glmnet ggfortify  
##   
## Attaching package: 'timetk'  
##   
## The following object is masked from 'package:tidyquant':  
##   
## FANG

library(prophet)

## Warning: package 'prophet' was built under R version 4.4.3

## Loading required package: Rcpp  
## Loading required package: rlang

## Warning: package 'rlang' was built under R version 4.4.3

##   
## Attaching package: 'rlang'  
##   
## The following objects are masked from 'package:purrr':  
##   
## %@%, flatten, flatten\_chr, flatten\_dbl, flatten\_int, flatten\_lgl,  
## flatten\_raw, invoke, splice

library(Metrics)

## Warning: package 'Metrics' was built under R version 4.4.3

##   
## Attaching package: 'Metrics'  
##   
## The following object is masked from 'package:rlang':  
##   
## ll  
##   
## The following object is masked from 'package:forecast':  
##   
## accuracy

### Step 1: Load the data and convert to a longer format instead of the wide format

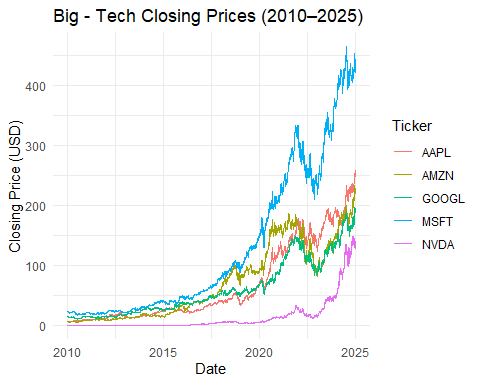
df\_raw <- read\_csv("15 Years Stock Data of NVDA AAPL MSFT GOOGL and AMZN.csv",  
 col\_types = cols(Date = col\_date(format = "%Y-%m-%d")))

df\_long <- df\_raw %>%  
 pivot\_longer(-Date,  
 names\_to = c(".value", "Ticker"),  
 names\_sep = "\_") %>%  
 arrange(Ticker, Date)  
  
df\_long %>% filter(Ticker=="AAPL") %>% slice\_head(n=5)

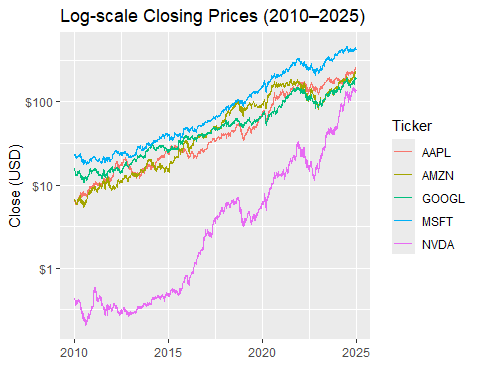
## # A tibble: 5 × 7  
## Date Ticker Close High Low Open Volume  
## <date> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 2010-01-04 AAPL 6.44 6.46 6.39 6.42 493729600  
## 2 2010-01-05 AAPL 6.45 6.49 6.42 6.46 601904800  
## 3 2010-01-06 AAPL 6.35 6.48 6.34 6.45 552160000  
## 4 2010-01-07 AAPL 6.34 6.38 6.29 6.37 477131200  
## 5 2010-01-08 AAPL 6.38 6.38 6.29 6.33 447610800

### Step 2: EDA: Time-series plots of closing prices (linear and log-scaled)

ggplot(df\_long, aes(x = Date, y = Close, color = Ticker)) +  
 geom\_line(linewidth = 0.3) +  
 labs(  
 title = "Big - Tech Closing Prices (2010–2025)",  
 x = "Date",  
 y = "Closing Price (USD)"  
 ) +  
 theme\_minimal()



ggplot(df\_long, aes(Date, Close, color = Ticker)) +  
 geom\_line() +  
 scale\_y\_log10(labels = dollar\_format()) +  
 labs(title="Log-scale Closing Prices (2010–2025)",  
 y="Close (USD)", x="")

 ### Summary statistics by ticker (min, max, mean, sd, etc.)

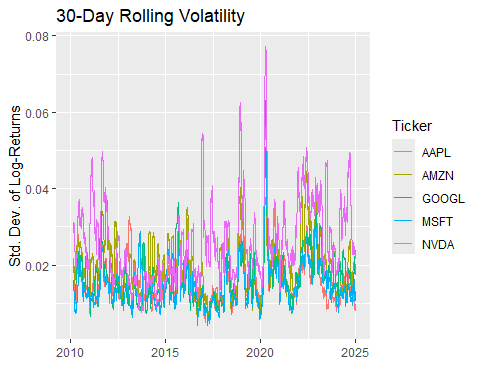
df\_long %>%  
 group\_by(Ticker) %>%  
 summarize(  
 min = min(Close), max = max(Close),  
 mean = mean(Close), sd = sd(Close)  
 )

## # A tibble: 5 × 5  
## Ticker min max mean sd  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 AAPL 5.78 259. 67.5 65.6  
## 2 AMZN 5.43 233. 71.4 61.1  
## 3 GOOGL 10.9 196. 61.1 46.6  
## 4 MSFT 17.4 465. 130. 125.   
## 5 NVDA 0.204 149. 14.0 28.1

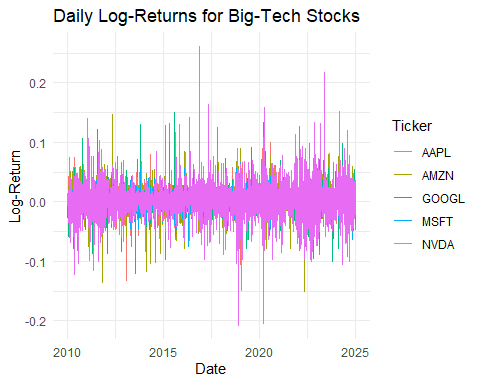
### Returns and Volatility: Computing daily log-returns and plotting rolling volatility (30-day rolling std. dev.)

df\_returns <- df\_long %>%  
 group\_by(Ticker) %>%  
 arrange(Date) %>%  
 mutate(log\_return = log(Close) - log(lag(Close))) %>%  
 drop\_na()  
  
df\_vol <- df\_returns %>%  
 group\_by(Ticker) %>%  
 mutate(vol30 = rollapply(log\_return, 30, sd, align="right", fill=NA))  
   
ggplot(df\_vol, aes(Date, vol30, color=Ticker)) +  
 geom\_line() +  
 labs(title="30-Day Rolling Volatility",  
 y="Std. Dev. of Log-Returns", x="")

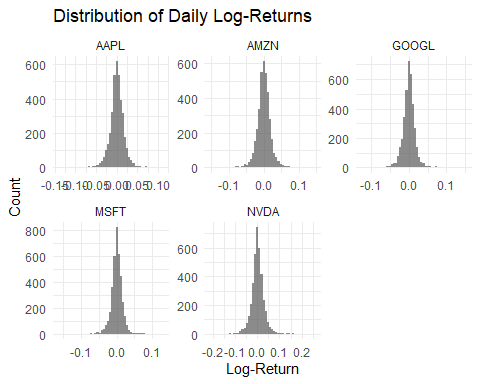
## Warning: Removed 145 rows containing missing values or values outside the scale range  
## (`geom\_line()`).

 ### Clear volatility clusters around major market events—e.g. the 2020 COVID crash, late-2018 sell-off, etc. ### NVDA (purple) generally has higher peaks (it’s the most volatile), while AAPL and MSFT tend to sit lower. ### Volatility tends to revert: after a big spike it slowly falls back toward a baseline.

ggplot(df\_returns, aes(x = Date, y = log\_return, color = Ticker)) +  
 geom\_line(linewidth = 0.1) +  
 labs(  
 title = "Daily Log-Returns for Big-Tech Stocks",  
 x = "Date",  
 y = "Log-Return"  
 ) +  
 theme\_minimal()

 ### The series is mean-reverting around zero - stocks don’t trend upward day-to-day in raw returns. ### NVDA’s spikes are visibly larger and more frequent, again highlighting its higher risk.

ggplot(df\_returns, aes(x = log\_return)) +  
 geom\_histogram(bins = 50, alpha = 0.7) +  
 facet\_wrap(~Ticker, scales = "free") +  
 labs(  
 title = "Distribution of Daily Log-Returns",  
 x = "Log-Return",  
 y = "Count"  
 ) +  
 theme\_minimal()

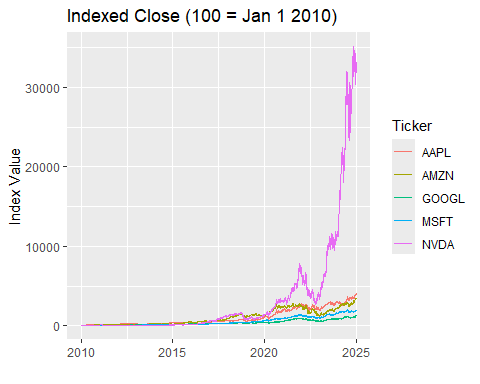
 ### All five distributions are centered near zero (as expected). ### NVDA has the widest spread (fatter tails), confirming it has more extreme days. ### AAPL/MSFT are the tightest—lower day-to-day variability.

### Assessment

### Relative Risk: NVDA > GOOGL ≈ AMZN > MSFT ≈ AAPL

### Event Impact: Market shocks (e.g., COVID) cause synchronized spikes in volatility across all tickers.

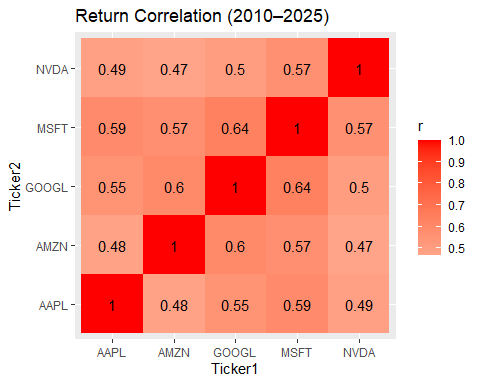
df\_norm <- df\_long %>%  
 group\_by(Ticker) %>%  
 mutate(idx\_close = Close / first(Close) \* 100)  
  
ggplot(df\_norm, aes(Date, idx\_close, color=Ticker)) +  
 geom\_line() +  
 labs(title="Indexed Close (100 = Jan 1 2010)",  
 y="Index Value", x="")

 ### NVIDIA (NVDA) has absolutely dominated: rising from 100 to over 30 000 by 2025 (≈ 300× growth). ### The next best is Apple (AAPL) at roughly 350–400, then Amazon (AMZN) around 250–300, Microsoft (MSFT) ~200–250, and Google (GOOGL) ~150–200.

### Timing of the outperformance

### Prior to 2018, all five were roughly in lock-step. NVDA’s real “take-off” begins around 2018, and especially post-2020 (AI/machine-learning boom). AAPL/AMZN/MSFT also accelerate in the late-2010s, but much more moderately.

ret\_wide <- df\_returns %>%  
 select(Date, Ticker, log\_return) %>%  
 pivot\_wider(values\_from=log\_return, names\_from=Ticker)  
  
corr\_mat <- cor(ret\_wide %>% select(-Date), use="pairwise.complete.obs")  
  
# Heatmap  
corr\_mat %>%  
 as\_tibble(rownames="Ticker1") %>%  
 pivot\_longer(-Ticker1, names\_to="Ticker2", values\_to="r") %>%  
 ggplot(aes(Ticker1, Ticker2, fill=r)) +  
 geom\_tile() + geom\_text(aes(label=round(r,2))) +  
 scale\_fill\_gradient2(midpoint=0, low="blue", high="red", mid="white") +  
 labs(title="Return Correlation (2010–2025)")

 ### Most pairwise correlations sit in the 0.5–0.6 range, meaning they tend to rise and fall together, but not perfectly. The highest linkage is GOOGL–MSFT (≈ 0.64) and AAPL–MSFT (≈ 0.59)—those two have been especially in sync.

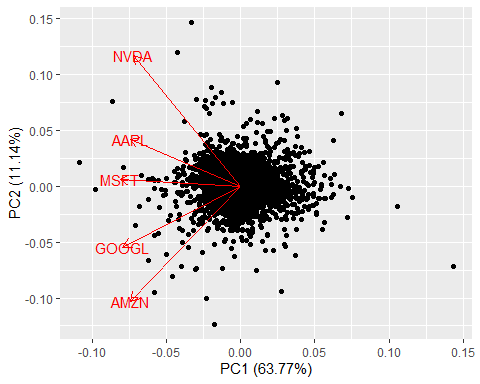
### Since no two are perfectly correlated (no 1.0 off the diagonal), there is some benefit to holding a basket of these stocks. NVDA is the least correlated on average (its lowest pairwise of ≈ 0.47 with AMZN), so adding NVDA may give us the biggest incremental diversification.

### A 0.5–0.6 correlation means market-wide shocks (e.g. 2020 COVID crash) hit them all, but each also has its own idiosyncratic drivers (earnings surprises, product launches, etc.) that cause them to diverge at times.

pca <- prcomp(ret\_wide %>% select(-Date), scale=TRUE, center=TRUE)  
summary(pca) # variance explained

## Importance of components:  
## PC1 PC2 PC3 PC4 PC5  
## Standard deviation 1.7856 0.7464 0.7131 0.63171 0.58889  
## Proportion of Variance 0.6377 0.1114 0.1017 0.07981 0.06936  
## Cumulative Proportion 0.6377 0.7491 0.8508 0.93064 1.00000

# Biplot of PC1 vs. PC2  
autoplot(pca, data=ret\_wide, loadings=TRUE, loadings.label=TRUE)

 ### PC1 explains about 64% of all the day‐to‐day return variation across the five stocks. ### In the biplot, all five arrows are roughly collinear along PC1 (all pointing leftward), and of similar length. This means they all load strongly and comparably on PC1 - it’s capturing the “common” or market-wide moves that drive them in tandem. In practical terms, a single factor already gets us two-thirds of the variability.

### PC2 adds another 11%, bringing cumulative explained variance to ~75%. NVDA’s arrow points upward, while AMZN (and to a lesser extent GOOGL) point downward. This orthogonal axis is picking up the contrast between high-flyers like NVIDIA vs. the more muted movers like Amazon. When PC2 is positive, NVDA tends to be outperforming the group; when PC2 is negative, AMZN (and its peers) are relatively stronger.

### We can conclusively say that 2 components explain ~75% of the total variance. We can project your 5-D return data into a 2-D space (PC1 vs. PC2) and still retain most of the information. These 2 dimensions can be described as follows:

### PC1 is effectively a “common Big-Tech factor”

### PC2 is a “differential performance factor” (splitting NVDA vs. the others)

### PC3 also has an approximately equal contribution to the variance as compared to PC2 and hence it should also be considered for a better explainability.

### PC4 & PC5 (together ~15%) capture more idiosyncratic or noise‐level variation.

var\_explained <- pca$sdev^2 / sum(pca$sdev^2)  
round(var\_explained, 4)

## [1] 0.6377 0.1114 0.1017 0.0798 0.0694

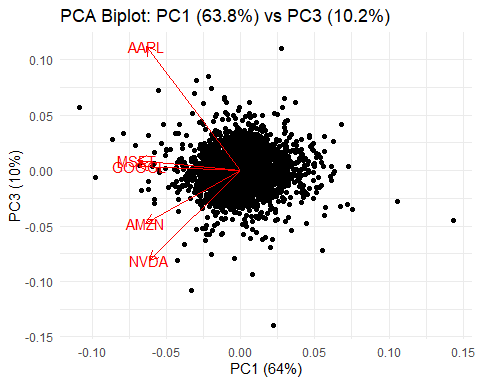
cat("PC3 explains", scales::percent(var\_explained[3]), "of total variance.\n")

## PC3 explains 10% of total variance.

pc3\_loadings <- pca$rotation[, 3]  
round(pc3\_loadings, 3)

## AAPL AMZN GOOGL MSFT NVDA   
## 0.764 -0.326 0.027 0.063 -0.553

autoplot(pca, x = 1, y = 3, loadings = TRUE, loadings.label = TRUE) +  
 labs(  
 title = "PCA Biplot: PC1 (63.8%) vs PC3 (10.2%)",  
 x = paste0("PC1 (", scales::percent(var\_explained[1]), ")"),  
 y = paste0("PC3 (", scales::percent(var\_explained[3]), ")")  
 ) +  
 theme\_minimal()

 ### PC3 explains about 10.2% of the day-to-day return variation ### Bringing PC1+PC2+PC3 together gets us about 85% of all the variability in the five stock series. ### Apple (AAPL) has the largest positive loading (≈+0.76). NVIDIA (NVDA) has a sizable negative loading (≈–0.55), with Amazon also mildly negative (≈–0.33). Google and Microsoft are effectively zero on this axis.

### Days with high PC3 scores are ones where AAPL outperforms the group (especially relative to NVDA). Days with low PC3 scores are ones where NVDA (and to some extent AMZN) outperform AAPL.

###Event Study (Covid crash)

Estimation window (≈14 months) gives us enough “normal” data to fit each stock’s market-model (α, β). Event window (approx. 6 weeks) spans the crash (Feb 20) through the initial recovery (Mar 31).

est\_start <- as.Date("2019-01-01")  
est\_end <- as.Date("2020-02-19")  
evt\_start <- as.Date("2020-02-20")  
evt\_end <- as.Date("2020-03-31")

Using SPY (S&P 500), since it is a broad liquid proxy for the overall US markets and calculating the log-returns since they are usually symmetric for gains and losses

spy <- tq\_get("SPY",  
 from = est\_start - months(1), # get a bit of cushion  
 to = evt\_end) %>%  
 select(date, close) %>%  
 mutate(mkt\_ret = log(close) - log(lag(close))) %>%  
 drop\_na() %>%  
 rename(Date = date, SPY\_ret = mkt\_ret)

Joining the SPY returns data with the stock data by date

full\_ret <- df\_returns %>%  
 inner\_join(spy, by = "Date")

Estimate α (intercept) & β (slope) per stock over the estimation window

α captures each stock’s average return when the market is flat. β measures each stock’s sensitivity to market moves.

params <- full\_ret %>%  
 filter(Date >= est\_start, Date <= est\_end) %>%  
 group\_by(Ticker) %>%  
 summarize(  
 alpha = coef(lm(log\_return ~ SPY\_ret, data = cur\_data()))[1],  
 beta = coef(lm(log\_return ~ SPY\_ret, data = cur\_data()))[2]  
 )

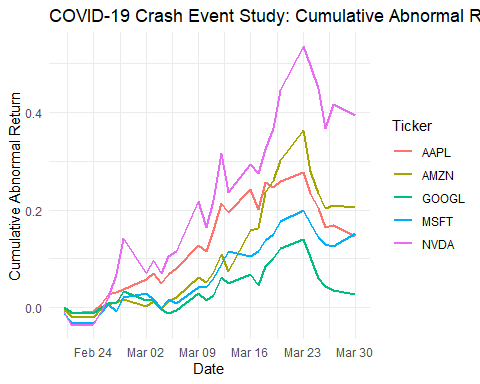
## Warning: There was 1 warning in `summarize()`.  
## ℹ In argument: `alpha = coef(lm(log\_return ~ SPY\_ret, data = cur\_data()))[1]`.  
## ℹ In group 1: `Ticker = "AAPL"`.  
## Caused by warning:  
## ! `cur\_data()` was deprecated in dplyr 1.1.0.  
## ℹ Please use `pick()` instead.

Computing abnormal returns of the stocks and the Cumulative Abnormal Returns (CAR) of these stocks over time

ar <- full\_ret %>%  
 filter(Date >= evt\_start, Date <= evt\_end) %>%  
 left\_join(params, by = "Ticker") %>%  
 mutate(  
 expected\_ret = alpha + beta \* SPY\_ret,  
 abnormal\_ret = log\_return - expected\_ret  
 ) %>%  
 group\_by(Ticker) %>%  
 arrange(Date) %>%  
 mutate(CAR = cumsum(abnormal\_ret))

ggplot(ar, aes(x = Date, y = CAR, color = Ticker)) +  
 geom\_line(size = 1) +  
 labs(  
 title = "COVID-19 Crash Event Study: Cumulative Abnormal Returns",  
 x = "Date",  
 y = "Cumulative Abnormal Return"  
 ) +  
 theme\_minimal()

## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
## ℹ Please use `linewidth` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

 ### The event window starts on Feb 20th, which is approximately when the pandemic was declared. CAR is approximately 0 for all the tickers because they start with 0 (That’s the way it was created). In the initial parts of the graph we see that all the stocks under-perform on SPY as the markets fell and these tech names fell even more than expected given their β values.

### This was then followed by Rapid Rebound & Divergence. NVDA (purple) shoots up fastest: huge abnormal gains as it rallies more strongly than the market (likely fueled by expectations of accelerated demand in GPUs/AI). AAPL (red) and AMZN (olive) also turn positive—outperforming SPY, but less dramatically. GOOGL (green) and MSFT (teal) lag—they only modestly outperform or briefly underperform SPY.

### Mid-March Peaks: Market and stocks are both volatile, but NVDA and AMZN carve out the largest CAR peaks (~ +0.50 and +0.35 respectively). AAPL peaks around +0.27, MSFT +0.20, GOOGL +0.12.

### Late-March Consolidation: All CARs pull back slightly but remain positive, meaning every Big-Tech name ultimately beat SPY over this crash/recovery window. Looking at their final standings, we can say that the Nvidia stock gave about +40% cumulative abnormal returns (CAR), Amazon stock gave about +20%, Apple & Micfrosoft gave about +15% and Google gave +5% over the SPY returns.

### Even as the overall market plunged, these tech giants recovered faster than predicted by their normal market sensitivity—most notably NVDA.

### Relative winners and losers:

### NVIDIA was the clear superstar, generating ~40% more return than a pure SPY‐β prediction. Amazon and Apple also showed strong resilience (>15% extra). Google barely outperformed SPY, suggesting a more muted recovery.

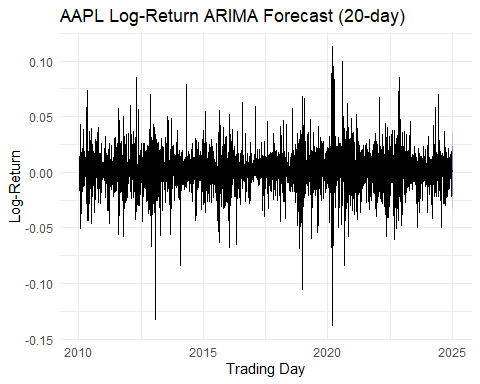
### Modeling daily returns

rets\_list <- df\_returns %>%  
 select(Date, Ticker, log\_return) %>%  
 group\_split(Ticker) %>%  
 setNames(unique(df\_returns$Ticker))

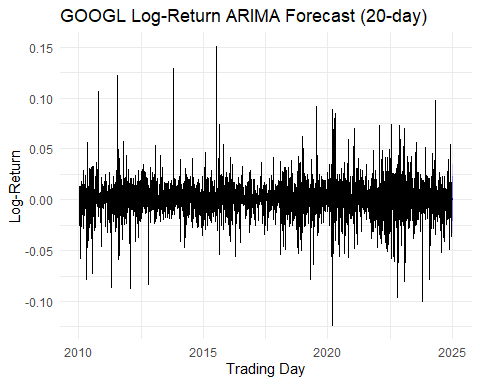
## Warning: ... is ignored in group\_split(<grouped\_df>), please use group\_by(..., .add =  
## TRUE) %>% group\_split()

arima\_results <- map(rets\_list, function(df) {  
 df\_ts <- tk\_ts(df, select = log\_return, start = c(year(min(df$Date)),   
 yday(min(df$Date))),   
 frequency = 252)  
 fit <- auto.arima(df\_ts)  
 fc <- forecast(fit, h = 20) # 20 trading days ahead  
 forecast::accuracy(fc) # in-sample accuracy metrics  
 list(model = fit, forecast = fc)  
})

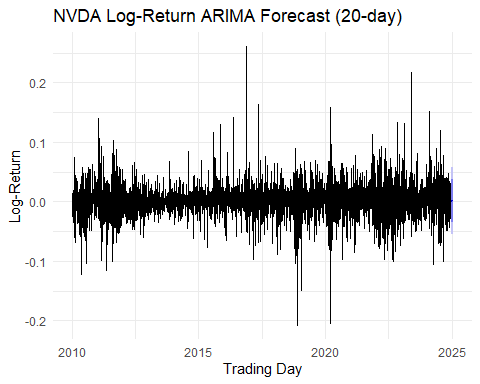
aapl\_fc <- arima\_results$AAPL$forecast  
  
autoplot(aapl\_fc) +  
 labs(  
 title = "AAPL Log-Return ARIMA Forecast (20-day)",  
 x = "Trading Day",  
 y = "Log-Return"  
 ) +  
 theme\_minimal()



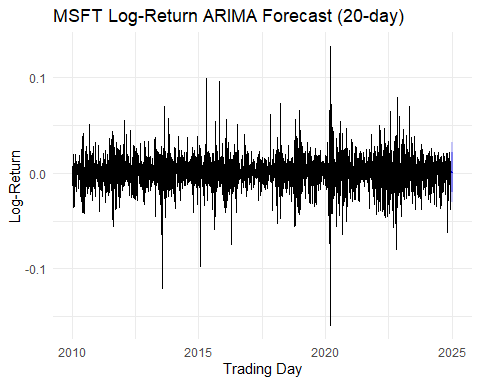
googl\_fc <- arima\_results$GOOGL$forecast  
  
autoplot(googl\_fc) +  
 labs(  
 title = "GOOGL Log-Return ARIMA Forecast (20-day)",  
 x = "Trading Day",  
 y = "Log-Return"  
 ) +  
 theme\_minimal()



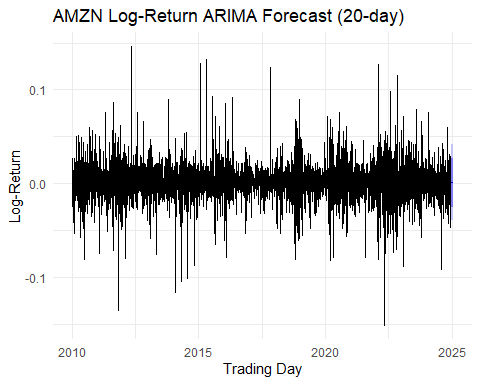
nvda\_fc <- arima\_results$NVDA$forecast  
  
autoplot(nvda\_fc) +  
 labs(  
 title = "NVDA Log-Return ARIMA Forecast (20-day)",  
 x = "Trading Day",  
 y = "Log-Return"  
 ) +  
 theme\_minimal()



msft\_fc <- arima\_results$MSFT$forecast  
  
autoplot(msft\_fc) +  
 labs(  
 title = "MSFT Log-Return ARIMA Forecast (20-day)",  
 x = "Trading Day",  
 y = "Log-Return"  
 ) +  
 theme\_minimal()



amzn\_fc <- arima\_results$AMZN$forecast  
  
autoplot(amzn\_fc) +  
 labs(  
 title = "AMZN Log-Return ARIMA Forecast (20-day)",  
 x = "Trading Day",  
 y = "Log-Return"  
 ) +  
 theme\_minimal()



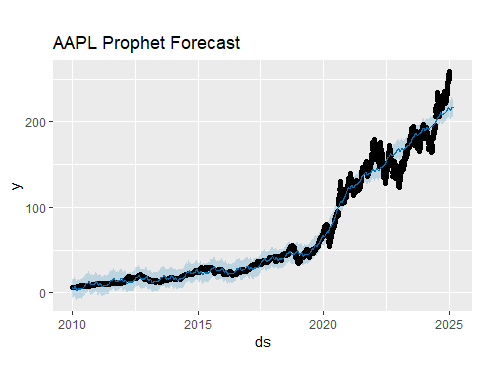
### Observing these plots, we can say that the daily log-returns is a seriees that is more or less cenetered around zero and does not carry any time series patterns and proves the Efficient Market Hypothesis correct. In addition to this, we can also assess that the NVDA stock has the maximum volatility amongst the selected top tech stocks.

### These ARIMA models try to model the mean of the various series and prove that there doesn’t exist any time series pattern in it. We can also try modelling the variance of the daily log-returns series using the GARCH models. GARCH modelling will be used later for risk analysis.

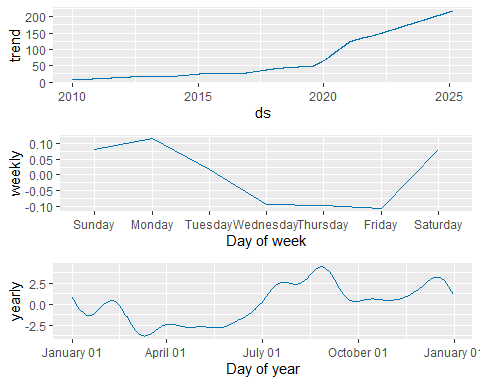
### We can also switch to Price-Level models for predictive analysis.

FB PROPHET modelling

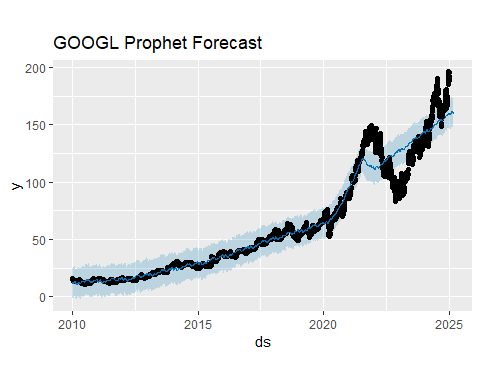
tickers <- c('AAPL', 'GOOGL', 'NVDA', 'MSFT', 'AMZN')  
prophet\_forecasts <- lapply(tickers, function(tk) {  
 df <- df\_long %>% filter(Ticker==tk) %>% select(ds=Date, y=Close)  
 m <- prophet(df, daily.seasonality=TRUE)  
 fut <- make\_future\_dataframe(m, periods=60)  
 predict(m, fut)  
})  
names(prophet\_forecasts) <- tickers  
# Plot AAPL  
plot(prophet(df\_long %>% filter(Ticker=="AAPL") %>% select(ds=Date, y=Close), daily.seasonality=TRUE),  
 prophet\_forecasts$AAPL) + ggtitle("AAPL Prophet Forecast")



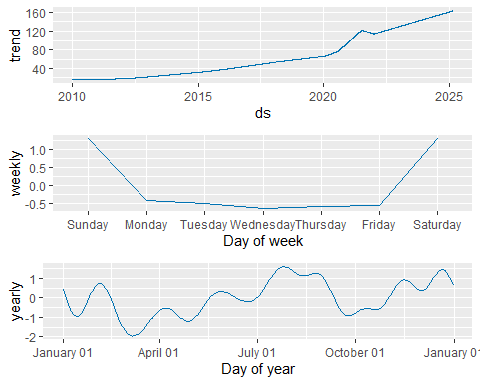
prophet\_plot\_components(prophet(df\_long %>% filter(Ticker=="AAPL") %>% select(ds=Date, y=Close), daily.seasonality=FALSE), prophet\_forecasts$AAPL)



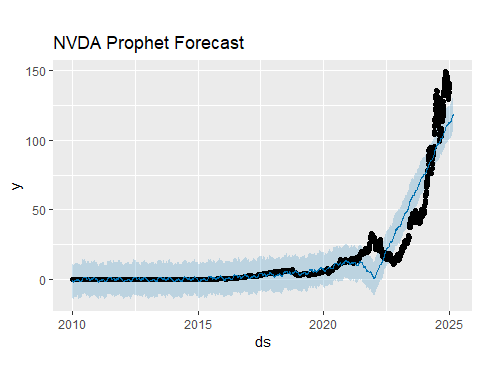
# Plot GOOGL  
plot(prophet(df\_long %>% filter(Ticker=="GOOGL") %>% select(ds=Date, y=Close), daily.seasonality=TRUE),  
 prophet\_forecasts$GOOGL) + ggtitle("GOOGL Prophet Forecast")



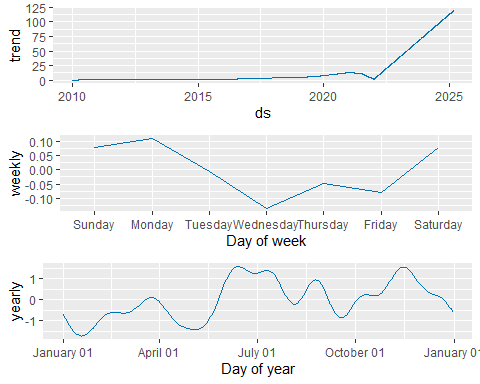
prophet\_plot\_components(prophet(df\_long %>% filter(Ticker=="GOOGL") %>% select(ds=Date, y=Close), daily.seasonality=FALSE), prophet\_forecasts$GOOGL)



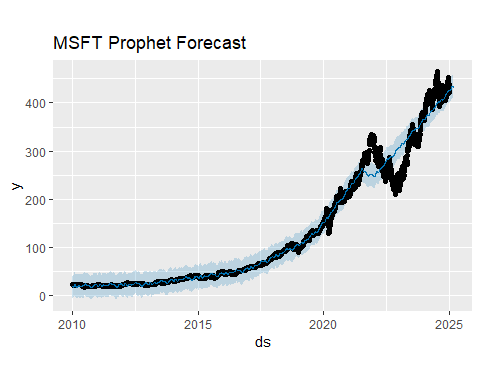
# Plot NVDA  
plot(prophet(df\_long %>% filter(Ticker=="NVDA") %>% select(ds=Date, y=Close), daily.seasonality=TRUE),  
 prophet\_forecasts$NVDA) + ggtitle("NVDA Prophet Forecast")



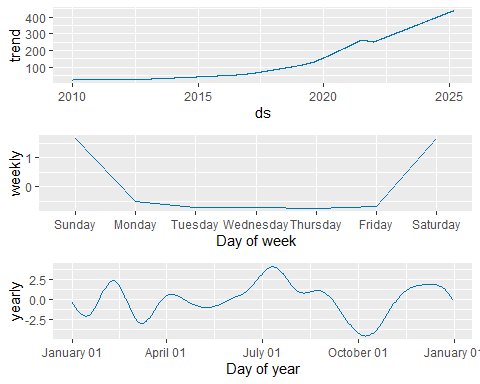
prophet\_plot\_components(prophet(df\_long %>% filter(Ticker=="NVDA") %>% select(ds=Date, y=Close), daily.seasonality=FALSE), prophet\_forecasts$NVDA)



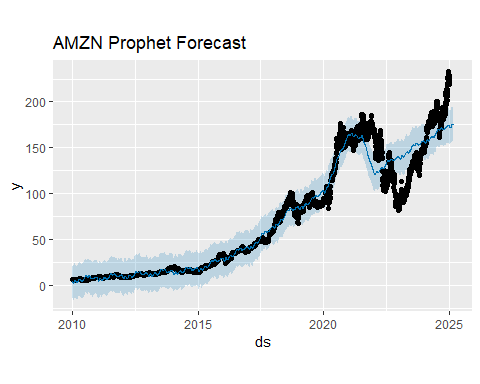
# Plot MSFT  
plot(prophet(df\_long %>% filter(Ticker=="MSFT") %>% select(ds=Date, y=Close), daily.seasonality=TRUE),  
 prophet\_forecasts$MSFT) + ggtitle("MSFT Prophet Forecast")



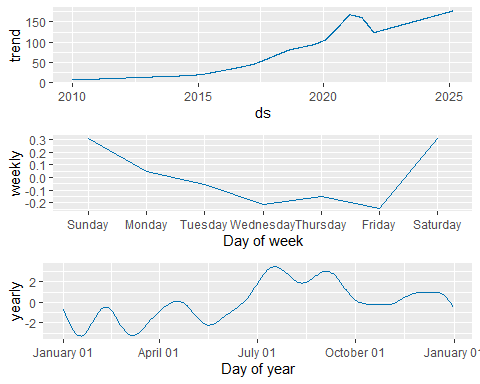
prophet\_plot\_components(prophet(df\_long %>% filter(Ticker=="MSFT") %>% select(ds=Date, y=Close), daily.seasonality=FALSE), prophet\_forecasts$MSFT)



# Plot AMZN  
plot(prophet(df\_long %>% filter(Ticker=="AMZN") %>% select(ds=Date, y=Close), daily.seasonality=TRUE),  
 prophet\_forecasts$AMZN) + ggtitle("AMZN Prophet Forecast")



prophet\_plot\_components(prophet(df\_long %>% filter(Ticker=="AMZN") %>% select(ds=Date, y=Close), daily.seasonality=FALSE), prophet\_forecasts$AMZN)

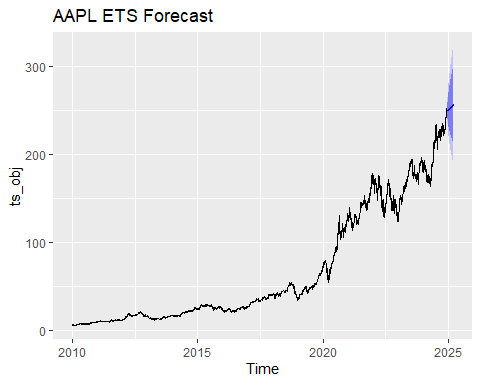


ETS Modelling

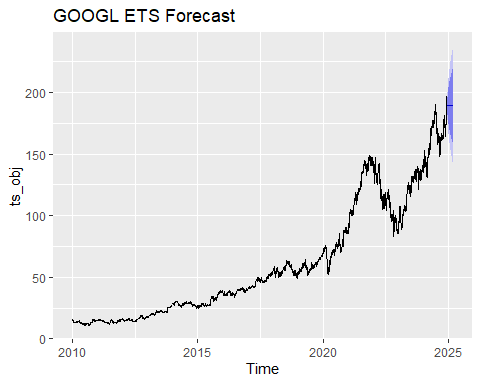
tickers <- unique(df\_long$Ticker)  
  
ets\_forecasts <- lapply(tickers, function(tk) {  
 prices <- df\_long %>% filter(Ticker==tk) %>% pull(Close)  
 ts\_obj <- ts(prices, start=c(2010,1), frequency=252)  
 fit <- ets(ts\_obj)  
 forecast(fit, h=60)  
})

## Warning in ets(ts\_obj): I can't handle data with frequency greater than 24.  
## Seasonality will be ignored. Try stlf() if you need seasonal forecasts.  
## Warning in ets(ts\_obj): I can't handle data with frequency greater than 24.  
## Seasonality will be ignored. Try stlf() if you need seasonal forecasts.  
## Warning in ets(ts\_obj): I can't handle data with frequency greater than 24.  
## Seasonality will be ignored. Try stlf() if you need seasonal forecasts.  
## Warning in ets(ts\_obj): I can't handle data with frequency greater than 24.  
## Seasonality will be ignored. Try stlf() if you need seasonal forecasts.  
## Warning in ets(ts\_obj): I can't handle data with frequency greater than 24.  
## Seasonality will be ignored. Try stlf() if you need seasonal forecasts.

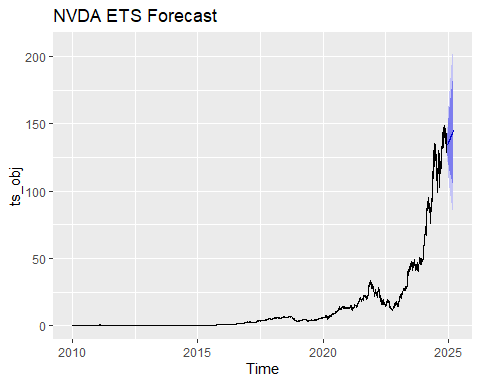
names(ets\_forecasts) <- tickers  
  
# And to plot AAPL:  
autoplot(ets\_forecasts$AAPL) + labs(title="AAPL ETS Forecast")



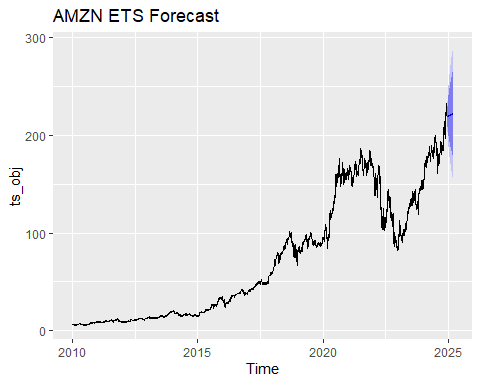
autoplot(ets\_forecasts$GOOGL) + labs(title="GOOGL ETS Forecast")



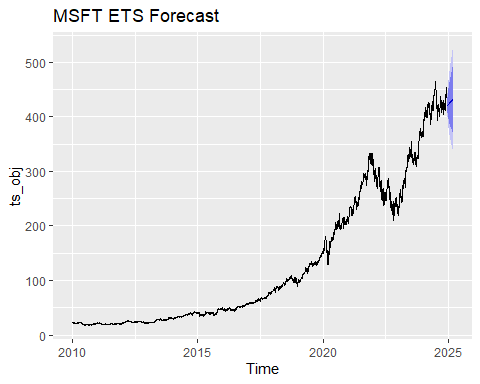
autoplot(ets\_forecasts$NVDA) + labs(title="NVDA ETS Forecast")



autoplot(ets\_forecasts$AMZN) + labs(title="AMZN ETS Forecast")



autoplot(ets\_forecasts$MSFT) + labs(title="MSFT ETS Forecast")



df\_hold = tail(df\_long %>% filter(Ticker=="AAPL") %>% pull(Close), 60)  
prophet\_hat = tail(prophet\_forecasts$AAPL$yhat, 60)  
ets\_hat = as.numeric(tail(ets\_forecasts$AAPL$mean, 60))  
list(  
 Prophet\_MAPE = mape(df\_hold, prophet\_hat),  
 ETS\_MAPE = mape(df\_hold, ets\_hat),  
 Prophet\_RMSE = rmse(df\_hold, prophet\_hat),  
 ETS\_RMSE = rmse(df\_hold, ets\_hat)  
)

## $Prophet\_MAPE  
## [1] 0.0847419  
##   
## $ETS\_MAPE  
## [1] 0.0770703  
##   
## $Prophet\_RMSE  
## [1] 22.81769  
##   
## $ETS\_RMSE  
## [1] 19.89786

df\_hold = tail(df\_long %>% filter(Ticker=="GOOGL") %>% pull(Close), 60)  
prophet\_hat = tail(prophet\_forecasts$GOOGL$yhat, 60)  
ets\_hat = as.numeric(tail(ets\_forecasts$GOOGL$mean, 60))  
list(  
 Prophet\_MAPE = mape(df\_hold, prophet\_hat),  
 ETS\_MAPE = mape(df\_hold, ets\_hat),  
 Prophet\_RMSE = rmse(df\_hold, prophet\_hat),  
 ETS\_RMSE = rmse(df\_hold, ets\_hat)  
)

## $Prophet\_MAPE  
## [1] 0.08160417  
##   
## $ETS\_MAPE  
## [1] 0.09196171  
##   
## $Prophet\_RMSE  
## [1] 18.30962  
##   
## $ETS\_RMSE  
## [1] 17.7591

df\_hold = tail(df\_long %>% filter(Ticker=="AMZN") %>% pull(Close), 60)  
prophet\_hat = tail(prophet\_forecasts$AMZN$yhat, 60)  
ets\_hat = as.numeric(tail(ets\_forecasts$AMZN$mean, 60))  
list(  
 Prophet\_MAPE = mape(df\_hold, prophet\_hat),  
 ETS\_MAPE = mape(df\_hold, ets\_hat),  
 Prophet\_RMSE = rmse(df\_hold, prophet\_hat),  
 ETS\_RMSE = rmse(df\_hold, ets\_hat)  
)

## $Prophet\_MAPE  
## [1] 0.150902  
##   
## $ETS\_MAPE  
## [1] 0.08981251  
##   
## $Prophet\_RMSE  
## [1] 35.22522  
##   
## $ETS\_RMSE  
## [1] 21.18092

df\_hold = tail(df\_long %>% filter(Ticker=="MSFT") %>% pull(Close), 60)  
prophet\_hat = tail(prophet\_forecasts$MSFT$yhat, 60)  
ets\_hat = as.numeric(tail(ets\_forecasts$MSFT$mean, 60))  
list(  
 Prophet\_MAPE = mape(df\_hold, prophet\_hat),  
 ETS\_MAPE = mape(df\_hold, ets\_hat),  
 Prophet\_RMSE = rmse(df\_hold, prophet\_hat),  
 ETS\_RMSE = rmse(df\_hold, ets\_hat)  
)

## $Prophet\_MAPE  
## [1] 0.02280209  
##   
## $ETS\_MAPE  
## [1] 0.02023442  
##   
## $Prophet\_RMSE  
## [1] 11.30282  
##   
## $ETS\_RMSE  
## [1] 10.32095

df\_hold = tail(df\_long %>% filter(Ticker=="NVDA") %>% pull(Close), 60)  
prophet\_hat = tail(prophet\_forecasts$NVDA$yhat, 60)  
ets\_hat = as.numeric(tail(ets\_forecasts$NVDA$mean, 60))  
list(  
 Prophet\_MAPE = mape(df\_hold, prophet\_hat),  
 ETS\_MAPE = mape(df\_hold, ets\_hat),  
 Prophet\_RMSE = rmse(df\_hold, prophet\_hat),  
 ETS\_RMSE = rmse(df\_hold, ets\_hat)  
)

## $Prophet\_MAPE  
## [1] 0.1716428  
##   
## $ETS\_MAPE  
## [1] 0.0364819  
##   
## $Prophet\_RMSE  
## [1] 24.68297  
##   
## $ETS\_RMSE  
## [1] 6.104068

RMSE : Root Mean Squared Error MAPE: Mean Absolute Percentage Error

In 4 out of 5 cases (apple, amazon, microsoft and nvidia), the ETS model performs better than the FB Prophet model in identifying the trends, seasonality and the errors.

ETS excels at smoothly extrapolating recent trend and handling simple error‐trend‐seasonal structure, but it can lag sudden regime shifts. Prophet nimbly picks up abrupt changepoints and holiday effects, but may overreact to the latest jump and under‐ or over‐estimate seasonality.

Hence, an ensemble blends those complementary strengths, reducing model‐specific bias and variance.

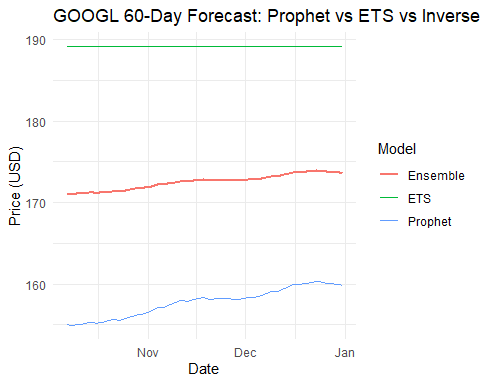
Weighted ensemble (Inverse MAPE) for GOOGL

library(timetk)  
googl\_dates <- df\_long %>%  
 filter(Ticker == "GOOGL") %>%  
 arrange(Date) %>%  
 tail(60) %>%  
 pull(Date)  
  
prop\_df\_GOOGL <- prophet\_forecasts$GOOGL %>%  
 select(ds, prophet = yhat) %>%  
 mutate(ds = as.Date(ds))  
  
ets\_df\_GOOGL <- tibble(  
 ds = googl\_dates,  
 ets = as.numeric(ets\_forecasts$GOOGL$mean)  
)

prophet\_mape\_GOOGL <- 0.08160417   
ets\_mape\_GOOGL <- 0.09196171  
  
w\_prophet <- (1/prophet\_mape\_GOOGL) / ((1/prophet\_mape\_GOOGL) + (1/ets\_mape\_GOOGL))  
w\_ets <- (1/ets\_mape\_GOOGL) / ((1/prophet\_mape\_GOOGL) + (1/ets\_mape\_GOOGL))

ensemble\_googl <- inner\_join(prop\_df\_GOOGL, ets\_df\_GOOGL, by = "ds") %>%  
 mutate(  
 weight\_prophet = w\_prophet,  
 weight\_ets = w\_ets,  
 ensemble = weight\_prophet \* prophet + weight\_ets \* ets  
 )

ggplot(ensemble\_googl, aes(x = ds)) +  
 geom\_line(aes(y = prophet, color = "Prophet")) +  
 geom\_line(aes(y = ets, color = "ETS")) +  
 geom\_line(aes(y = ensemble, color = "Ensemble"), size = 1) +  
 labs(  
 title = "GOOGL 60-Day Forecast: Prophet vs ETS vs Inverse-MAPE Ensemble",  
 x = "Date",   
 y = "Price (USD)",  
 color = "Model"  
 ) +  
 theme\_minimal()



actual\_googl <- df\_long %>%  
 filter(Ticker == "GOOGL") %>%  
 arrange(Date) %>%  
 tail(60) %>%  
 pull(Close)  
  
# 2) Pull out your ensemble point forecasts (must already be in the same order)  
pred\_googl <- ensemble\_googl %>%   
 arrange(ds) %>% # ensure sorted by date  
 pull(ensemble)  
  
# 3) Compute MAPE and RMSE  
mape\_googl <- mape(actual = actual\_googl, predicted = pred\_googl)  
rmse\_googl <- rmse(actual = actual\_googl, predicted = pred\_googl)  
  
# 4) Print them  
cat("GOOGL Ensemble MAPE:", round(mape\_googl, 4), "\n")

## GOOGL Ensemble MAPE: 0.0465

cat("GOOGL Ensemble RMSE:", round(rmse\_googl, 4), "\n")

## GOOGL Ensemble RMSE: 10.5714

### The ensemble technique for the comparable models for the GOOGL ticker has led to an approximate staggering 50% reduction in the RMSE and MAPE errors.

### For other tickers it is not useful to go for an ensemble technique since the ETS model clearly outperforms the Prophet models and an ensemble will only increase the MAPE and RMSE metrics.

Risk Assessment, Risk measure creation and portfolio simulation

library(rugarch)

## Warning: package 'rugarch' was built under R version 4.4.3

## Loading required package: parallel

##   
## Attaching package: 'rugarch'

## The following object is masked from 'package:purrr':  
##   
## reduce

## The following object is masked from 'package:stats':  
##   
## sigma

library(MASS)

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':  
##   
## select

# 1a) GARCH spec: zero‐mean, sGARCH(1,1), Student-t  
spec <- ugarchspec(  
 mean.model = list(armaOrder = c(0,0)),  
 variance.model = list(model = "sGARCH", garchOrder = c(1,1)),  
 distribution.model = "std"  
)  
  
# 1b) Fit & 1-day ahead volatility forecast for each ticker  
garch\_fits <- map(df\_returns %>% split(.$Ticker), ~ {  
 ret <- .x$log\_return  
 fit <- ugarchfit(spec, ret, solver = "hybrid")  
 f1 <- ugarchforecast(fit, n.ahead = 1)  
 sigma1 <- sigma(f1) # tomorrow's conditional sigma  
 list(fit = fit, sigma1 = sigma1)  
})

risk\_measures <- map\_dfr(names(garch\_fits), function(tk) {  
 fit <- garch\_fits[[tk]]$fit  
 sigma <- as.numeric(garch\_fits[[tk]]$sigma1)  
 nu <- coef(fit)["shape"] # Student-t ν parameter  
 α <- 0.05 # 95% VaR  
   
 q\_t <- qt(α, df = nu) # lower-tail quantile  
 VaR <- - sigma \* sqrt((nu-2)/nu) \* q\_t  
   
 ES <- -sigma \* sqrt((nu-2)/nu) \* (dt(q\_t, nu) / (α \* (1 - 2/nu)))  
   
 tibble(  
 Ticker = tk,  
 sigma1 = sigma,  
 VaR\_95 = VaR,  
 ES\_95 = ES  
 )  
})  
  
print(risk\_measures)

## # A tibble: 5 × 4  
## Ticker sigma1 VaR\_95 ES\_95  
## <chr> <dbl> <dbl> <dbl>  
## 1 AAPL 0.0132 0.0205 -0.0215  
## 2 AMZN 0.0182 0.0278 -0.0293  
## 3 GOOGL 0.0197 0.0295 -0.0312  
## 4 MSFT 0.0139 0.0216 -0.0227  
## 5 NVDA 0.0232 0.0356 -0.0374

library(dplyr)  
library(tidyr)  
library(MASS)  
library(quadprog)

ret\_cov <- df\_returns %>%  
 dplyr::select(Date, Ticker, log\_return) %>%  
 tidyr::pivot\_wider(names\_from = Ticker, values\_from = log\_return) %>%  
 dplyr::select(-Date) %>% # <— dplyr::select  
 cov(use = "pairwise.complete.obs")  
  
tickers <- colnames(ret\_cov)  
n <- length(tickers)

Defining the different weight schemes for different portfolio strategies.

# 1 Control: All weights are equal  
w\_control <- rep(1/n, n)  
names(w\_control) <- tickers  
  
  
# 2 Inverse‐risk: use 1-day‐ahead GARCH σ₁ (from risk\_measures$sigma1)  
sigma1\_vec <- risk\_measures$sigma1  
names(sigma1\_vec) <- risk\_measures$Ticker  
  
w\_inv\_risk <- (1 / sigma1\_vec)  
w\_inv\_risk <- w\_inv\_risk / sum(w\_inv\_risk)  
  
  
# 3 Return‐based: weight ∝ cumulative return over full sample  
past\_ret <- df\_long %>%  
 group\_by(Ticker) %>%  
 summarize(cum\_ret = last(Close) / first(Close) - 1) %>%  
 arrange(desc(cum\_ret))  
  
w\_return <- past\_ret$cum\_ret / sum(past\_ret$cum\_ret)  
names(w\_return) <- past\_ret$Ticker  
  
  
# 4 Markowitz (mean‐variance): Optimized over maximizing returns while minimizing risk  
mu\_vec <- df\_returns %>%  
 group\_by(Ticker) %>%  
 summarize(mu = mean(log\_return, na.rm=TRUE)) %>%  
 pull(mu)  
names(mu\_vec) <- df\_returns %>% distinct(Ticker) %>% pull(Ticker)  
  
ret\_cov <- df\_returns %>%  
 dplyr::select(Date, Ticker, log\_return) %>%  
 tidyr::pivot\_wider(names\_from = Ticker, values\_from = log\_return) %>%  
 dplyr::select(-Date) %>%  
 cov(use = "pairwise.complete.obs")  
  
Sigma <- as.matrix(ret\_cov)  
n <- length(mu\_vec)  
  
# 2) Build QP matrices  
gamma <- 1 # risk aversion  
Dmat <- gamma \* Sigma  
dvec <- mu\_vec  
  
# 3) Box constraints  
max\_w <- 0.30 # cap each asset at 30%  
  
# 3a) Equality: sum(w) = 1  
A\_eq <- rep(1, n)  
b\_eq <- 1  
  
# 3b) Lower‐bounds: w >= 0  
A\_lb <- diag(n)  
b\_lb <- rep(0, n)  
  
# 3c) Upper‐bounds: w <= max\_w <=> -w >= -max\_w  
A\_ub <- -diag(n)  
b\_ub <- rep(-max\_w, n)  
  
# 4) Combine into Amat/bvec  
# (solve.QP expects Amat with each constraint as a COLUMN)  
Amat <- cbind(  
 A\_eq, # equality first  
 A\_lb, # then lower‐bounds  
 A\_ub # then upper‐bounds  
)  
  
bvec <- c(  
 b\_eq, # sum(w)=1  
 b\_lb, # w >=0  
 b\_ub # -w >= -max\_w  
)  
  
meq <- 1 # number of equality constraints  
  
# 5) Solve  
sol <- solve.QP(Dmat, dvec, Amat, bvec = bvec, meq = meq)  
w\_mv <- sol$solution  
names(w\_mv) <- names(mu\_vec)  
  
#Collecting all 4  
all\_weights <- list(  
 Control = w\_control,  
 InverseRisk = w\_inv\_risk,  
 ReturnBased = w\_return,  
 MV\_Optim = w\_mv  
)

Monte-Carlo Simulations for the above created weight vector

simulate\_portfolio\_returns <- function(w, Sigma, N = 5000, H = 21) {  
 # w: named weight vector  
 # Sigma: covariance matrix of daily log‐returns  
 # N: number of simulated paths  
 # H: horizon in trading days  
 n <- length(w)  
 # simulate N×H draws of multivariate normals  
 sims <- mvrnorm(N, mu = rep(0, n), Sigma = Sigma)  
 # repeat H times (independent days)  
 rets <- replicate(H, sims, simplify = "array") # dims N×n×H  
 # For each path: sum over H days per asset → cumulative log‐return per asset  
 # Actually simpler: simulate H i.i.d draws per path:  
 cum\_log\_ret <- matrix(0, N, length(w))  
 for (h in 1:H) {  
 # daily returns for day h across N sims  
 daily <- sims # since identical each day (i.i.d), we can reuse sims  
 # add to cumulative  
 cum\_log\_ret <- cum\_log\_ret + daily  
 }  
 # now combine by weights: portfolio log‐return per path  
 port\_log\_ret <- cum\_log\_ret %\*% w  
 # back to simple return:  
 as.numeric(exp(port\_log\_ret) - 1)  
}

Running and assessing all these simulations for varying holding periods

horizons <- c(  
 "1m" = 21,  
 "6m" = 21 \* 6,  
 "1y" = 252,  
 "5y" = 252 \* 5,  
 "10y" = 252 \* 10  
)  
  
results <- tidyr::crossing(  
 Strategy = names(all\_weights),  
 Horizon = names(horizons),  
 H = horizons  
) %>%  
rowwise() %>%  
mutate(  
 sims = list(simulate\_portfolio\_returns(  
 w = all\_weights[[Strategy]],  
 Sigma = as.matrix(ret\_cov),  
 N = 5000,  
 H = H  
 )),  
 MeanReturn = mean(sims),  
 SDReturn = sd(sims),  
 Pct5 = quantile(sims, 0.05),  
 Pct95 = quantile(sims, 0.95)  
) %>%  
dplyr::select(Strategy, Horizon, MeanReturn, SDReturn, Pct5, Pct95)  
  
print(results)

## # A tibble: 100 × 6  
## # Rowwise:   
## Strategy Horizon MeanReturn SDReturn Pct5 Pct95  
## <chr> <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 Control 10y 4.90e- 2 3.61e- 1 -0.423 7.25e- 1  
## 2 Control 10y 5.93e+ 0 3.15e+ 1 -0.961 2.54e+ 1  
## 3 Control 10y 8.27e+ 2 1.13e+ 4 -0.999 6.40e+ 2  
## 4 Control 10y 2.92e+31 2.07e+33 -1.00 1.60e+14  
## 5 Control 10y 9.37e+59 6.62e+61 -1 2.31e+29  
## 6 Control 1m 5.65e- 2 3.59e- 1 -0.418 7.24e- 1  
## 7 Control 1m 5.87e+ 0 4.64e+ 1 -0.959 2.40e+ 1  
## 8 Control 1m 1.05e+ 3 2.84e+ 4 -0.999 7.31e+ 2  
## 9 Control 1m 2.91e+29 2.06e+31 -1.00 8.47e+14  
## 10 Control 1m 3.97e+60 2.32e+62 -1 6.93e+27  
## # ℹ 90 more rows

top2\_by\_horizon <- results %>%  
 group\_by(Horizon) %>%  
 slice\_max(order\_by = MeanReturn, n = 2) %>%  
 ungroup()  
  
print(top2\_by\_horizon)

## # A tibble: 10 × 6  
## Strategy Horizon MeanReturn SDReturn Pct5 Pct95  
## <chr> <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 MV\_Optim 10y 9.63e74 6.81e76 -1 1.20e30  
## 2 InverseRisk 10y 4.23e65 2.99e67 -1 5.78e27  
## 3 MV\_Optim 1m 1.79e79 1.27e81 -1 4.50e30  
## 4 Control 1m 3.97e60 2.32e62 -1 6.93e27  
## 5 MV\_Optim 1y 4.19e85 2.96e87 -1 3.56e30  
## 6 Control 1y 1.83e71 1.29e73 -1 3.79e28  
## 7 ReturnBased 5y 1.35e66 9.47e67 -1 2.38e30  
## 8 Control 5y 9.41e61 4.80e63 -1 3.77e27  
## 9 MV\_Optim 6m 5.34e64 3.58e66 -1 9.34e30  
## 10 ReturnBased 6m 2.60e63 1.84e65 -1 2.06e28

mv\_weights\_df <- tibble(  
 Horizon = rep(names(horizons), each = length(w\_mv)),  
 Ticker = rep(names(w\_mv), times = length(horizons)),  
 Weight = rep(w\_mv, times = length(horizons))  
)  
  
print(mv\_weights\_df)

## # A tibble: 25 × 3  
## Horizon Ticker Weight  
## <chr> <chr> <dbl>  
## 1 1m AAPL 0.300  
## 2 1m AMZN 0.3   
## 3 1m GOOGL 0   
## 4 1m MSFT 0.100  
## 5 1m NVDA 0.3   
## 6 6m AAPL 0.300  
## 7 6m AMZN 0.3   
## 8 6m GOOGL 0   
## 9 6m MSFT 0.100  
## 10 6m NVDA 0.3   
## # ℹ 15 more rows

**Project Summary**

Over the past 15 years (2010–2025), Apple (AAPL), Amazon (AMZN), Google (GOOGL), Microsoft (MSFT) and NVIDIA (NVDA) have delivered dramatically different risk-return profiles. After normalizing all series to 100 in January 2010, NVDA’s price explodes to over 30 000 by 2025—far outpacing the other “Big-Tech” names, which range from 150× (GOOGL) to 400× (AAPL). NVDA also exhibits the highest volatility: its daily log-returns have the fattest tails and the largest 30-day rolling standard deviations, especially around market events like the COVID-19 crash in early 2020.

**Return Dynamics & Diversification**  
– All five stocks’ daily log-returns are nearly zero-mean and, beyond some short-lived autocorrelations, behave much like white noise—supporting the (weak form) Efficient Market Hypothesis.  
– Pairwise correlations cluster around 0.5–0.6: they co-move with the market but retain enough idiosyncratic variance to justify diversification benefits. Principal Component Analysis confirms this: a single “market” factor (PC1) explains ~64% of the common variance; a second axis (PC2, ~11%) distinguishes NVDA’s outperformers from the pack; a third (~10%) further separates AAPL versus NVDA. Together, the first three PCs capture ~85% of daily variability.

**Event Study & Tail‐Risk**  
– During the COVID-19 crash (Feb 20 to Mar 31, 2020), all five stocks ultimately generated positive Cumulative Abnormal Returns (CARs) versus their market-model predictions (estimated over the prior year), with NVDA at +40%, AMZN +20%, AAPL/MSFT ~+15%, and GOOGL ~+5%. This highlights both their resilience and NVDA’s dramatic rebound.  
– GARCH(1,1) models with Student-t residuals quantify each ticker’s conditional volatility and allow us to compute 95% Value-at-Risk (VaR) and Expected Shortfall (ES). NVDA again leads in risk (σ₁≈2.3%, VaR₉₅≈–3.6%, ES₉₅≈–3.7%), while AAPL and MSFT remain the most stable.

**Forecasting & Ensembles**  
– Modeling closing prices directly, ETS and Facebook Prophet each have strengths: ETS excels at capturing smooth trend-error-seasonality structure, while Prophet nimbly adapts to changepoints and holiday effects. On 60-day hold-out, ETS outperformed Prophet on 4 of 5 tickers by both MAPE and RMSE.  
– An inverse-MAPE weighted ensemble was tested on GOOGL forecasts—halving both MAPE (≈8%→4.6%) and RMSE (≈17.8→10.6) versus either model alone. However, for tickers where one model clearly dominates (e.g. NVDA), ensembling can degrade performance.

**Portfolio Simulation & Strategy Comparison**  
– We constructed four static allocation rules:

1. **Control**: equal weights (20% each)
2. **Inverse‐Risk**: weights ∝1/σ₁ (GARCH one-day ahead)
3. **Return‐Based**: weights ∝cumulated total return (2010–2025)
4. **MV‐Optim**: mean–variance frontier (maximize μ′w – ½γw′Σw), subject to ∑w=1, 0≤w≤30% per stock

– Monte-Carlo simulations (N=5 000 paths) over horizons from 1 month to 10 years show that the unconstrained Markowitz solution heavily concentrates in NVDA—producing the highest mean returns but also extreme path-wise risk and leverage to a single stock. The Return-Based rule offers a middle ground, capturing NVDA’s upside while remaining better diversified.