Search tool for possible undervalued Stocks with DCF and P/B and subsequent Stock evaluation

Justin König

20.08.2022

Contents

```
# Die folgenden Funktionen sind eigentlich dazu gedacht gesourced zu werden
# werden jedoch der vollständigkeitshalber hier aufgeführt

# extrahiert den Preis, die Industry, EPS (earnings per share) und den
# Firmennamen von Yahoo Finances und berechnet den PE (Price to earnings)

basics_data <- function(ticker) {

  # fügt der URL mit hilfe der paste0 funktion die Variable ticker hinzu
  # die 0 an paste bewirkt das weglassen von leerzeichen
  url_profile <- paste0('https://finance.yahoo.com/quote/', ticker)

  # weißt der Variablen den HTML Text der URL zu
  # hier werden der HTML Text des Profils der Firma hinzugefügt
  html_profile <- read_html(url_profile) %>% html_node('body') %>%
    html_text() %>%
    toString()

  # extrahiert den HTML Text der financials Seite von Yahoo Finance der Firma
  url_fin <- paste0('https://finance.yahoo.com/quote/',
                    ticker, '/financials?p=', ticker)
  html_fin <- read_html(url_fin) %>% html_node('body') %>%
    html_text() %>% toString()

  # extrahiert aus dem HTML Text den aktuellen Börsenpreis der Firma
  # die qdapRegex Funktion extrahiert hierbei alle Textpassagen zwischen den
  # Schlagwörtern. Da die Zahlen aus dem Text als nicht numerisch extrahiert
  # wird, müssen diese im FOlgenden immer in numerische charactere umgewandelt
  # werden.
  # Im Anschluss wird die Zahl auf zwei Nachkommastellen gerundet
  Price <- qdapRegex::ex_between(html_profile, "currentPrice\":{\"raw\":", ",\"fmt\":\"")[[1]]
  Price <- as.numeric(Price)
  Price <- round(Price, 2)

  # Extrahiert den Subsektor der Firma, da mehrere Textpassagen extrahiert werden
```

```
  # wird nur der erste genutzt (die gewollte Industry)
  Industry <- qdapRegex::ex_between(html_profile, "industry\":\"", "\"")[[1]]
  Industry <- Industry[1]

  # extrahiert die Währung
  Currency <- qdapRegex::ex_between(html_fin, "financialCurrency\":\"", "\"},\"price")[[1]]
  Currency <- data.frame(Currency)

  # extrahiert den EPS

  EPS <- qdapRegex::ex_between(html_profile, "EPS (TTM)", "Earnings")[[1]]
  EPS <- EPS[1]
  EPS <- as.numeric(EPS)

  # berechnen den PE und rundet auf zwei Nachkommastellen

  PE <- round(Price/EPS,2)

  # extrahiert den Firmennamen

  Company <- qdapRegex::ex_between(html_profile, "\":{\"title\":\"", " (")[[1]]
  Company <- Company[1]

  # weißt einer Liste die Variablen Price, Industry, PE, and Company zu und
  # gibt diese aus

  out <- list (
    Price = Price,
    Industry = Industry,
    PE = PE,
    Company = Company,
    Currency = Currency
  )
  return(out)
}
```

```
# eine while Schleife, nimmt den Ticker und weißt diesem die korrekte Industry,
# den Preis und P/E, sowie den Firmennamen zu
# in Klammern steht toassaign_df, da bei nutzen der Funktion, dieser zugewiesen
# werden kann

basics_assignement <- function(toassaign_df) {

  # x <- 1, damit die while Schleife in der ersten Reihe des data frames
  # beginnt zu zählen. Erstellt einen data frame aus toassaign_df, Industry,
  # Price und P/E und füllt die Reihen mit den Variablen
  x<-1
  Company <- "Company"
  Company <- data.frame(Company)
  toassaign_df <- bind_cols(Company, toassaign_df)
  Industry<-"Industry"
  Industry<-data.frame(Industry)
  toassaign_df<- bind_cols(toassaign_df, Industry)
```

```
  Currency<-"Currency"
  Currency<-data.frame(Currency)
  toassaign_df<- bind_cols(toassaign_df, Currency)
  Price<-"Price"
  Price<-data.frame(Price)
  toassaign_df<- bind_cols(toassaign_df, Price)
  PE <- "P/E"
  PE <- data.frame(PE)
  toassaign_df<- bind_cols(toassaign_df, PE)

  # entfernt die Variablen, damit sie mit attach() genutz werden können
  rm(Company,Industry,Price,PE, Currency)

  # es wird eine while schleife erstellt, welche von der ersten Reihe (x=1) bis
  # zur letzten Reihe (nrow) des data frames zählt
  while (x<=nrow(toassaign_df)) {
    # der aktuelle Ticker wird zugewiesen und die function basics_data wird
    # angewended um die benötigten datan des tickers zu scrapen
    ticker <- toassaign_df[paste(x),2]
    basics <- basics_data(paste(ticker))

    # attach ermöglicht hier den direkten zugriff auf die col Namen
    attach(basics)

    # Gescrapten Daten werden dem data frame zugewiesen
    # Zudem werden Statments gedruckt, welche den Nutzer wissen lässt,
    # an welcher Stelle die zuweisung aktuell ist
    toassaign_df[paste(x),1] <- Company
    print(paste("assining Industry", x, "of",nrow(toassaign_df), "to", Company))
    toassaign_df[paste(x),3] <- Industry
    print(paste("assining Currency", x, "of",nrow(toassaign_df), "to", Company))
    toassaign_df[paste(x),4] <- Currency
    print(paste("assining Price", x, "of",nrow(toassaign_df), "to", Company))
    toassaign_df[paste(x),5] <- Price
    print(paste("calculating and assining P/E", x, "of",nrow(toassaign_df), "to", Company))
    toassaign_df[paste(x),6] <- round(PE,2)

    # addiert x + 1 damit die nächste Reihe bearbeitet wird
    # detach wird angewandt, damit die daten nicht maskiert werden
    x<-x+1
    detach(basics)
  }
  # gibt eine Liste mit den gescrapten und zugewiesenen daten aus
  out <- list(
    list = toassaign_df
  )
  return(out)
}
```

```
# Die ergebnisse dieses chunks werden nicht in dem HTML Dokument ausgegeben,
# da bei vielen Tickern ein großer output entsteht

# Die Ticker der ausgesuchten Firmen, werden mittels eines Vektors der
```

```r
# Variablen Ticker zugewiesen

Ticker <-c("EOAN.DE", "GS", "MSFT", "AAPL", "BAC", "CRM", "NVDA", "NKE", "MS")

# Erstellt einen data frame aus Ticker
Stocks <- data.frame(Ticker)

# die Variable Ticker wird nicht mehr benötigt und wird entfernt
rm(Ticker)

# wendend die Funktion basics_assignment auf den data frane Stocks an, als
# Ticker der Funktion werden die in Klammern geschriebenen Stocks genutzt
# und nutzt anschliesend nut den Teil der Listen, welche benötigt wird
Stocks <- basics_assignement(Stocks)
Stocks <- Stocks$list
```

```r
# Erstellt eine Funktion, welche die benötigten Daten für die Berechnung des
# Fair Values der Aktien läd die benötigten Variablen sind die Ticker, sowie
# die TGR (Terminal Growth Rate) diese ist das angenommene konstante Wachstum
# mit welche das Unternehmen für immer wächst

DCF_data_scraper <- function(ticker, TGR = 0.025) {

  # es werden diverese URLs  von yahoo mit dem ticker verknüpft und anschließend
  # die HTML Text zugewiesen
  url_profile <- paste0('https://finance.yahoo.com/quote/', ticker)

  # Druckt Statments, damit der User weiß an welchem Punkt sich der scraper
  # befindet
  print("assingning urls and loading HTML")

  url_fin <- paste0('https://finance.yahoo.com/quote/',
                    ticker, '/financials?p=', ticker)
  html_fin <- read_html(url_fin) %>% html_node('body') %>%
    html_text() %>% toString()
  url_cf <- paste0('https://finance.yahoo.com/quote/',
                   ticker, '/cash-flow?p=', ticker)
  html_cf <- read_html(url_cf) %>% html_node('body') %>% html_text() %>% toString()
  url_bonds <- 'https://finance.yahoo.com/bonds'
  html_bonds <- read_html(url_bonds) %>%
    html_node('body') %>%  html_text() %>% toString()
  url_stats <- paste0('https://finance.yahoo.com/quote/',
                      ticker, '/key-statistics?p=', ticker)
  html_stats <- read_html(url_stats) %>% html_node('body') %>%
    html_text() %>% toString()
  url_balance <- paste0('https://finance.yahoo.com/quote/',
                        ticker, '/balance-sheet?p=', ticker)
  html_balance <- read_html(url_balance) %>% html_node('body') %>%
    html_text() %>% toString()
  url_ana <- paste0('https://finance.yahoo.com/quote/',
                    ticker,'/analysis?p=',ticker)
  html_ana <- read_html(url_ana) %>% html_node('body') %>%
    html_text() %>% toString()
```

```r
# URL für das equity risk premium
url_risk_prem<-read_html('https://pages.stern.nyu.edu/~adamodar/New_Home_Page/datafile/ctryprem.html')

# da es bei diesem HTML Text Probleme mit qdapRegex gibt wird das
# equity risk premiummit filfe eines HTML Nodes extrahiert, welcher die
# Tabelle auf der Website ausgibt
url_risk_prem <- url_risk_prem %>% html_nodes("td") %>%
  html_text()

# extrahiert die Jahre in welcher die Finacel Statments herrausgegeben wurden
# und erstellt einen data frame daraus
Year <- qdapRegex::ex_between(html_fin, "financialsChart\":{\"yearly\":", "quarterly\":[{\"")[[1]]
Year <- qdapRegex::ex_between(Year, "{\"date\":", ",\"revenue\"")[[1]]
yearly_data <- data.frame(Year)

# extrahiert den Revenue
print("extracting Revenue")

TotalRevenue <- qdapRegex::ex_between(html_fin, ",\"revenue\":{\"raw\":", ",\"fmt\":\"")[[1]]
TotalRevenue <- TotalRevenue[1:4]
TotalRevenue <- as.numeric(TotalRevenue)
TotalRevenue <- data.frame(TotalRevenue)

yearly_data <- bind_cols(yearly_data, TotalRevenue )

print("extracting Projected Revenue and number of Analysts")
# extrahiert den avg. prohected revenue für die kommenden 2 Jahre und die
# Anzahl der Analysten, welche diese Vorhersage getroffen haben

ProjRev <- qdapRegex::ex_between(html_ana, "revenueEstimate", "numberOfAnalysts")[[1]]
ProjRev <- ProjRev[3:4]
ProjRev <- c(qdapRegex::ex_between(ProjRev, "\"avg\":{\"raw\":", ",\"fmt")[[1]],
             qdapRegex::ex_between(ProjRev, "\"avg\":{\"raw\":", ",\"fmt")[[2]] )
ProjRev <- as.numeric(ProjRev)
ProjRev <- data.frame(ProjRev)

ProjAnalysts <- qdapRegex::ex_between(html_ana, "revenueEstimate", "yearAgoRevenue")[[1]]
ProjAnalysts <- ProjAnalysts[3:4]
ProjAnalysts <- c(qdapRegex::ex_between(ProjAnalysts, "\"numberOfAnalysts\":{\"raw\":", ",\"fmt")[[1]]
                  qdapRegex::ex_between(ProjAnalysts, "\"numberOfAnalysts\":{\"raw\":", ",\"fmt")[[2]]
ProjAnalysts <- as.numeric(ProjAnalysts)
ProjAnalysts <- data.frame(ProjAnalysts)
ProjRev <- bind_cols(ProjRev,ProjAnalysts)

# Extrahiert den EBIT
print("extracting EBIT")

EBIT <- qdapRegex::ex_between(html_fin, "annualOperatingIncome", "annual")[[1]]
EBIT <- qdapRegex::ex_between(EBIT, "{\"raw\":", ",\"fmt\"")[[1]]
EBIT <- as.numeric(EBIT[1:4])
EBIT <- data.frame(EBIT)
yearly_data <- bind_cols(yearly_data, EBIT)
```

```r
# Extrahier den Income Tax Expense
print("extracting income tax expense")

IncTaxEx <- qdapRegex::ex_between(html_balance, "\"incomeTaxExpense\":{\"raw\":", ",\"fmt\":")[[1]]
IncTaxEx <- rev(IncTaxEx[5:8])
IncTaxEx <- as.numeric(IncTaxEx)
IncTaxEx <- data.frame(IncTaxEx)
yearly_data <- bind_cols(yearly_data,IncTaxEx)

# extrahiert Depreciation & Amortization (D&A)
print("extracting Depreciation & Amortization")

DandA <- qdapRegex::ex_between(html_fin, "annualReconciledDepreciation", "trailing")[[1]]
DandA <- qdapRegex::ex_between(DandA, "\"raw\":", ",\"fmt\":")[[1]]
DandA <- DandA[1:4]
DandA <- as.numeric(DandA)
DandA <- data.frame(DandA)
yearly_data <- bind_cols(yearly_data,DandA)

print("extracting Capital Expenditure")

# extrahiert CAPEX (Capital Expenditure)
CAPEX <- qdapRegex::ex_between(html_cf, "\"capitalExpenditures\":", "\"fmt\"")[[1]]
CAPEX <- CAPEX[1:4]
CAPEX <- gsub("{\"raw\":", "", CAPEX, fixed=T)
CAPEX <- gsub(",", "", CAPEX, fixed=T)
CAPEX <- as.numeric(CAPEX)
CAPEX <- CAPEX*-1
CAPEX <- rev(CAPEX)
CAPEX <- data.frame(CAPEX)
yearly_data <- bind_cols(yearly_data, CAPEX)

# extrahiert NWC () Changein net working capital)
cNWC <- qdapRegex::ex_between(html_cf, "annualChangeInWorkingCapital", "trailing")[[1]]
cNWC <- qdapRegex::ex_between(cNWC, "\"raw\":", ",\"fmt\":")[[1]]
cNWC <-  cNWC[1:4]
cNWC <- as.numeric(cNWC)
cNWC <- data.frame(cNWC)
yearly_data <- bind_cols(yearly_data, cNWC)

# extrahiert das ende des Fiskaljahres des Unternehmens und berechnet den
# aktuellen Prozensatz für die später folgende calenderization
fye <- qdapRegex::ex_between(html_stats, "lastFiscalYearEnd", "heldPercentInstitutions")[[1]]
fyemonth <- qdapRegex::ex_between(fye, "-", "-")[[1]]
fyemonth <- as.numeric(fyemonth)
rateofffye <- fyemonth/12

# erstellent den data frame TOData (für aktuelle nicht wiederholende Daten)
TOData <- data.frame(rateofffye)

# extrahiert das Market Cap
print("extracting Market Cap")
```

```
 MarkCap <- qdapRegex::ex_between(html_stats, "trailingMarketCap", "\"fmt\":\"")[[1]]
 MarkCap <- qdapRegex::ex_between(MarkCap, "\"raw\":", ",")[[1]]
 MarkCap <- as.numeric(MarkCap)
 MarkCap <- data.frame(MarkCap)
 TOData <- bind_cols(TOData,MarkCap)

 # extrahiert die 10 Year Bondrate (R_f (risc free rate))
 print("extracting 10 Year Bond Rates")

 BR <- qdapRegex::ex_between(html_bonds, "Yield 10 Years", "0")[[1]]
 BR <- BR[1]
 BR <- as.numeric(BR)/100
 TOData <- data.frame(TOData, BR)

 # extrahiert die Beta
 print("extracting Beta")

 Beta <- qdapRegex::ex_between(html_stats, "(5Y Monthly)", "-")[[1]]
 Beta <- Beta[1]
 Beta <- as.numeric(Beta)
 Beta <- data.frame(Beta)
 TOData <- bind_cols(TOData,Beta)

# extrahiert das equity risk premium der USA
 print("extracting equity risk premium")

 erp <- url_risk_prem[1103]
 erp <- gsub("%", "", erp, fixed=T)
 erp <- as.numeric(erp)/100
 erp <- data.frame(erp)
 TOData <- bind_cols(TOData,erp)

 # extrahiert current Debt
 print("extracting current Debt")

 CurrDebt <- qdapRegex::ex_between(html_balance, "CurrentDebtAndCapitalLeaseObligation", "annual")[[1]]
 CurrDebt <- CurrDebt[2]
 CurrDebt <- qdapRegex::ex_between(CurrDebt, "\"raw\":", ",\"fmt\":\"")[[1]]
 CurrDebt <- rev(CurrDebt)
 CurrDebt <- CurrDebt[1]
 CurrDebt <- as.numeric(CurrDebt)
 CurrDebt <- data.frame(CurrDebt)
 TOData <- bind_cols(TOData,CurrDebt)

 # extrahiert die long term Debt
 print("extracting long term Debt")

 LongDebt <- qdapRegex::ex_between(html_balance, "longTermDebt\"", "inventory")[[1]]
 LongDebt <- qdapRegex::ex_between(LongDebt, "\"raw\":", ",\"fmt\":\"")[[1]]
 LongDebt <- LongDebt[1]
 LongDebt <- as.numeric(LongDebt)
 LongDebt <- data.frame(LongDebt)
 TOData <- bind_cols(TOData,LongDebt)
```

```r
# extrahiert die Interest Expense
print("extracting Interest Expense")

# da es bei Yahoo Finances bei unterschiedlichen Unternehmen dazu kommt,
# dass unterschiedliche Überschriften, für die gleichen Werte genutzt werden
# wird hier zunächst nach dem Standard gesucht, falls dieser nicht verfügbar
# ist, wird eine if Schleife genutzt.
# Diese extrahiert von einer anderen Stelle, falls der Wert der ersten
# extraktion NA ist
IntExpense <- qdapRegex::ex_between(html_fin, "annualInterestExpense", "annual")[[1]]
IntExpense <- qdapRegex::ex_between(IntExpense, "{\"raw\":", ",\"fmt\"")[[1]]
IntExpense <- rev(IntExpense)
IntExpense <- IntExpense[1]
IntExpense <- as.numeric(IntExpense)

if (is.na(IntExpense) == TRUE) {
  IntExpense <- qdapRegex::ex_between(html_fin, "NonOperatingInterestIncomeExpense", "annual")[[1]]
  IntExpense <- IntExpense[3]
  IntExpense <- qdapRegex::ex_between(IntExpense, "{\"raw\":", ",\"fmt\"")[[1]]
  IntExpense <- as.numeric(IntExpense)
  if (IntExpense < 0) {
    IntExpense <- IntExpense*-1
  }

}
IntExpense <- data.frame(IntExpense)
TOData <- bind_cols(TOData,IntExpense)

# extrahiert die ausstehenden Aktien
print("extracting Outstanding Shares")

floatshares <- qdapRegex::ex_between(html_stats, "\"floatShares\":{\"raw\":", ",\"fmt\"")[[1]]
floatshares <- as.numeric(floatshares)
floatshares <- data.frame(floatshares)
TOData <- bind_cols(TOData,floatshares)

# extrahiert die Steuern vor Einkommen
print("extracting pre income tax")

PreIncTax <- qdapRegex::ex_between(html_fin, "incomeBeforeTax\":{\"raw\":", ",\"fmt\"")[[1]]
PreIncTax <- PreIncTax[5]
PreIncTax <- as.numeric(PreIncTax)
PreIncTax <- data.frame(PreIncTax)
TOData <- bind_cols(TOData,PreIncTax)

# extrahiert das Cash Vermögen des Unternehmens
cash <- qdapRegex::ex_between(html_balance, "otherAssets", "total")[[1]]
cash <- qdapRegex::ex_between(html_balance, "\"cash\":{\"raw\":", ",\"fmt\":")[[1]]
cash <- cash[1]
cash <- as.numeric(cash)
cash <- data.frame(cash)
TOData <- bind_cols(TOData, cash)
```

```
  # Weißt dem data frame TO data, die oben eingegeben TGR zu
  TOData<-data.frame(TOData,TGR)

  out <- list(
    yearly_data = yearly_data,
    TOData = TOData,
    ProjRev = ProjRev
  )
  return(out)
}
```

```
# nimmt einen data frame und berechnet die calenderization für diesen
calenderization <- function(toassaign_df) {
  # nimmt die berechnetet rateofffye des DCF scrapers und weißt sie
  # einer neuen Variablen zu
  rateofffye <- scraped_DCF_data[[1]]$TOData$rateofffye

  # weißt der varibalen y den Wert 2 zu, damit später damit auf die zweite col
  # zugegriffen wird
   y<-2

  # wiederholt die erste while schleife, von 2 bis das ende der col Anzahl des
  # df erreicht ist.
  while (y<=ncol(toassaign_df)) {
  # weist x den Wert 1 zu, damit in der ersten Reihe begonnen wird zu zählen
  x <- 1
  # wiederholt die zweite Schleife bis das ende der Reihen erreicht ist

  while(x<=nrow(toassaign_df)){
  # variable a und b werden genutzt um die Rechnung zu verkleinern
  a <- toassaign_df[as.numeric(paste(x)),as.numeric(paste(y))]

  # addiert x + 1 damit die Daten der Nächsten reihe genutzt werden
  x <- x+1
  b <- toassaign_df[as.numeric(paste(x)),as.numeric(paste(y))]

  # berechnet die calenderized data mit Hilfe der rye
  calen <- a*rateofffye+b*(1-rateofffye)

  # x - 1 damit die gewünschte Zeile mit der calenderized data ersetzt wird
  x <- x-1
  toassaign_df[as.numeric(paste(x)),as.numeric(paste(y))] <- calen

  # addiert wieder x + 1 damit die Schleife mit der nächsten Reihe weiter rechnet
  x <- x+1
  }
  # wenn eine col mit der Hilfe der zweiten Schleife ersetzt wurde, wird
  # y + 1 addiert damit in der nächsten col weiter gerechnet wird
  y <- y+1

  }
  out <- list(
    list = toassaign_df
```

```
  )
  return(out)
}
```

```
# Funktion für die berechnung des DCFs

DCF_calculation <- function(yearly_data = scraped_DCF_data[[1]]$yearly_data,
                            TOData = scraped_DCF_data[[1]]$TOData,
                            ProjRev = scraped_DCF_data[[1]]$ProjRev$ProjRev) {

  last_statement <- as.numeric(yearly_data$Year[4])
  Year <- c(last_statement+1, last_statement+2, last_statement+3, last_statement+4,
            last_statement+5, last_statement+6)
  Projection <- data.frame(Year)

  # calculates the Revenue growth rate and takes the mean for further
  # calculations

  RevGrowthRate <- c((yearly_data$TotalRevenue[2]/yearly_data$TotalRevenue[1])-1,
                     (yearly_data$TotalRevenue[3]/yearly_data$TotalRevenue[2])-1,
                     (yearly_data$TotalRevenue[4]/yearly_data$TotalRevenue[3])-1,
                     (ProjRev[1]/yearly_data$TotalRevenue[4])-1,
                     (ProjRev[2]/ProjRev[1])-1)

  ## needs commentary

  meanRevGrowthRate <- mean(RevGrowthRate)
  calc_df <- data.frame(meanRevGrowthRate)

  EBITofRev <- c(yearly_data$EBIT[1:4]/yearly_data$TotalRevenue[1:4])
  meanEBITofRev <- mean(EBITofRev)
  meanEBITofRev <- data.frame(meanEBITofRev)
  calc_df <- bind_cols(calc_df,meanEBITofRev)

  TaxesofEBIT <- c(yearly_data$IncTaxEx[1:4]/yearly_data$EBIT[1:4])
  meanTaxesofEBIT <- mean(TaxesofEBIT)
  meanTaxesofEBIT <- data.frame(meanTaxesofEBIT)
  calc_df <- bind_cols(calc_df,meanTaxesofEBIT)

  DandAofRev <- c(yearly_data$DandA[1:4]/yearly_data$TotalRevenue[1:4])
  meanDandAofRev <- mean(DandAofRev)
  meanDandAofRev <- data.frame(meanDandAofRev)
  calc_df <- bind_cols(calc_df,meanDandAofRev)

  CAPEXofRev <- c(yearly_data$CAPEX[1:4]/yearly_data$TotalRevenue[1:4])
  meanCAPEXofRev <- mean(CAPEXofRev)
  meanCAPEXofRev <- data.frame(meanCAPEXofRev)
  calc_df <- bind_cols(calc_df,meanCAPEXofRev)

  cNWCofRev <- c(yearly_data$cNWC[1:4]/yearly_data$TotalRevenue[1:4])
  meancNWCofRev <- mean(cNWCofRev)
  meancNWCofRev <- data.frame(meancNWCofRev)
  calc_df <- bind_cols(calc_df,meancNWCofRev)
```

```r
# takes the Revenue growth rate and projects the future Revenue

rev3 <-  (ProjRev[2]*calc_df$meanRevGrowthRate+ProjRev[2])*calc_df$meanRevGrowthRate+(ProjRev[2]*
                              calc_df$meanRevGrowthRate+ProjRev[2])
rev4 <- rev3*calc_df$meanRevGrowthRate+rev3
rev5 <- rev4*calc_df$meanRevGrowthRate+rev4

Rev_Proj <- c(ProjRev,
              (ProjRev[2]*calc_df$meanRevGrowthRate+ProjRev[2]),
              rev3, rev4, rev5)
rm(rev3,rev4,rev5)

Rev_Proj <- data.frame(Rev_Proj)
Projection <- bind_cols(Projection,Rev_Proj)

EBIT_Proj <- c(Projection$Rev_Proj[1:6]*calc_df$meanEBITofRev)
EBIT_Proj <- data.frame(EBIT_Proj)
Projection <- bind_cols(Projection, EBIT_Proj)

Taxes_Proj <- c(Projection$EBIT_Proj[1:6]*calc_df$meanTaxesofEBIT)
Taxes_Proj <- data.frame(Taxes_Proj)
Projection <- bind_cols(Projection, Taxes_Proj)

DandA_Proj <- c(Projection$Rev_Proj[1:6]*calc_df$meanDandAofRev)
DandA_Proj <- data.frame(DandA_Proj)
Projection <- bind_cols(Projection, DandA_Proj)

CAPEX_Proj <- c(Projection$Rev_Proj[1:6]*calc_df$meanCAPEXofRev)
CAPEX_Proj <- data.frame(CAPEX_Proj)
Projection <- bind_cols(Projection, CAPEX_Proj)

cNWC_Proj <- c(Projection$Rev_Proj[1:6]*calc_df$meancNWCofRev)
cNWC_Proj <- data.frame(cNWC_Proj)
Projection <- bind_cols(Projection, cNWC_Proj)

#Calenderization of Data.frames

Projection <- Projection %>%
  rename(
    TotalRevenue = Rev_Proj,
      EBIT = EBIT_Proj,
    IncTaxEx = Taxes_Proj,
    DandA = DandA_Proj,
    CAPEX = CAPEX_Proj,
    cNWC = cNWC_Proj
  )

yearly_and_proj <- bind_rows(yearly_data[2:7], Projection[2:7])
Year <- c(as.numeric(yearly_data[,1]),Projection[,1])
Year <- data.frame(Year)
past_and_proj <- bind_cols(Year, yearly_and_proj)

calended_data <- calenderization(past_and_proj)
```

```
  calended_data <- calended_data$list
  calended_data <- slice(calended_data,1:(nrow(calended_data)-1))

  # calculates EBIAT (earnings before interes after taxes)
  # EBIT - Taxes

  EBIAT <- c(calended_data$EBIT[1:nrow(calended_data)]-
               calended_data$IncTaxEx[1:nrow(calended_data)])
  EBIAT <- data.frame(EBIAT)
  calended_data <- bind_cols(calended_data, EBIAT)


  FCF <- c(calended_data$EBIAT[1:9]+calended_data$DandA[1:9]-
             calended_data$CAPEX[1:9]-calended_data$cNWC[1:9])
  FCF <- data.frame(FCF)
  calended_data <- bind_cols(calended_data, FCF)


  # calculates left over days of the year for mid year convention
  # todays date from Sys.Date

  date <- format(Sys.Date())

  # defines the last day of the year

  end_year <- ceiling_date(Sys.Date() %m-% months(1), 'year') %m-% days(1)

  # calculates the days between currend date and end of the year

  days <- difftime(end_year, date)
  days <- as.numeric(days)
  pofyear <- round(days/365,2)
  midyear <- round(pofyear/2,2)

  DiscountYear <- c(rep("NA", 4), midyear, pofyear+0.5, pofyear+1.5, pofyear+2.5, pofyear+3.5)
  DiscountYear <- data.frame(DiscountYear)
  calended_data <- bind_cols(calended_data, DiscountYear)

  out <- list(
    calended_data = calended_data,
    calc_df = calc_df
  )
  return(out)
}
```

```
toassaign_df <- Stocks
  x<-1
  DCF_ISP <-"DCF_ISP"
  DCF_ISP <- data.frame(DCF_ISP)
  PB <- "P/B"
  PB <- data.frame(PB)
  toassaign_df<- bind_cols(toassaign_df, DCF_ISP)
  toassaign_df<- bind_cols(toassaign_df, PB)
```

“‘