```
## IPART's Equity beta model version 6.0
## This model will calculate an estimate of Beta for a selection of proxy firms.
## The model does this by using a list of proxy firms from a Stocks Import Template Excel file,
## stored in a user-specified working directory.  This file is available on our website as
## "Input into R model - Selection of Stocks - Public Template".
## The user will need to populate this Excel template with a list of proxy firms.
## For queries please contact Edward Jenkins at IPART on edward_jenkins@ipart.nsw.gov.au

#### ---- 00 - PREPARATION ----
## This clears the R environment.
rm(list=ls())

## The model also assumes that certain packages have been installed on the user's computer.
## If not, please uncomment the code below and run it to install packages used in this model.
## install.packages("DatastreamDSWS2R");install.packages("dplyr");install.packages("lubridate")
## install.packages("reshape2");install.packages("stringr");install.packages("broom");install.packages("tidyverse")
## install.packages("purrr");install.packages("readxl");install.packages("openxlsx");install.packages("ggplot2")

### END OF PREPARATION ##

#### ---- 01 - SETTING UP DATASTREAM ----
## This section of the model activates the datastream library and sets up the user credentials.
#Activating the library.
library(DatastreamDSWS2R)

#The user will need their own username and password for datastream.
#The user must enter these in quotation marks in the two lines below
options(Datastream.Username = "") #Username
options(Datastream.Password = "") #Password
dsws <- dsws$new()

### END OF SETTING UP DATASTREAM ##

#### ---- 02 - SETTING UP THE MODEL ----
```

```
## The required package libraries are activated and the working
directory is set.
# Loading package libraries that are needed to run the rest of
the script.
library(dplyr)
library(lubridate)
library(reshape2)
library(stringr)
library(broom)
library(tidyverse)
library(purrr)
library(readxl)
library(openxlsx)
library(ggplot2)

# Selecting and setting the working directory.
winDialog(type='ok','Please select working directory')
wd <- choose.dir()
setwd(wd)

### END OF SETTING UP THE MODEL ##

#### ---- 03 - SETTING THE CONSTANTS ----
#This opens a windows dialogue box to select the user's Stocks
Import Template Excel file
winDialog(type='ok','Please select Stocks Import Template Excel
file')
file <- choose.files()
#This imports the selected file
inp <- read_excel(file,sheet='Input')

#These next lines take the values from the imported file and make
them objects in the R environment.
coylist <- inp$Symbol
sDate <- inp$Start_Date[1]
sDatetm1 <- inp$Start_Date[2]
sDatetm2 <- inp$Start_Date[3]
sDatetm3 <- inp$Start_Date[4]
sDatetm4 <- inp$Start_Date[5]
eDate <- inp$End_Date[1]
amihud_cutoff <- inp$Amihud_Cutoff[1]
target_DE <- inp$Target_DE[1]
weeks <- inp$min_weeks[1]
min_days_traded <- inp$min_days_traded[1]
filename <- inp$file_name[1]

#This creates the working directory folder where all the outputs
will be saved
dir.create(paste0("folder_", filename))
#This creates an object with the same name as the new folder
```

```
dir.name <- paste0("/folder_", filename)
#this sets the working directory as the newly created folder
setwd(paste0(wd, dir.name))

## END OF SETTING THE CONSTANTS ##

#### ---- 04 - GETTING THE STATIC DATA ----
#This is a blank list that will contain the proxy firms static
data
tmp_coy_s = list()

#This loop selects each item on the proxy firms list and extracts
information from datastream.
#Each firms result is saved as a list in the tmp_coy_s list.
for (i in 1:length(coylist))
{ loopstatic      <- dsws$snapshotRequest(instrument =
coylist[i],
                                          datatype   =
c('NAME','ISIN','BETA','ESTAT', 'MI#MNEM','MNEM', 'ISINID'),
                                          requestDate = "0D")
tmp_coy_s[[length(tmp_coy_s)+1]] = loopstatic}

#This turns the list into a dataframe
coy_s_data <- bind_rows(tmp_coy_s, .id = "Instrument")

## A work around after "INDX" was renamed to "MI#NEM"
# The code changes "MI#NEM" (current name) to "INDX" (previous
name), to keep remainder of code unchanged
colnames(coy_s_data)[7] <- "INDX"

#This removes firms that have NA as a result (cannot be found in
datastream)
coy_s_data <- coy_s_data[coy_s_data[,"ISIN"]!="NA",]
#This gets rid of the instruments that are a secondary security
coy_s_data <- coy_s_data[coy_s_data[,"ISINID"]!="S",]
#This gets rid of the instruments that have no matching market
coy_s_data <- coy_s_data[coy_s_data[,"INDX"]!="NA",]

#This makes a list made up of firms' ISIN (International
Securities Identification Number) values
coylist <- coy_s_data$ISIN

#there are two columns called "instrument", so this removes the
first instance
coy_s_data[1] <- NULL

## END OF GETTING THE STATIC DATA ##

#### ---- 05 - GETTING THE FIRM RETURN DATA ----
### The process of getting the weekly firm return data is
```

```
repeated 5 times.
### This is repeated 5 times to ensure it returns data for each
possible week start and end combination,
### ie. Mon-Fri, Tue-Mon, Wed-Tue, Thu-Wed, Fri-Thu, referred to
as t-1 to t-4 (or tm1 to tm4) in the model.

##GETTING THE FIRM DATA FOR PERIOD t ##

#This creates a blank list that this iteration of weekly firm
return data will be saved into.
tmp_coy_RI = list()

#This loop takes each item on the firm list and then gets weekly
firm return data from datastream.
#Results are saved as a list on the tmp_coy_RI list.
for (i in 1:length(coylist))
{ loopRI      <- dsws$timeSeriesRequest(instrument = coylist[i],
                                        datatype   = "RI",
                                        startDate  = sDate,
                                        endDate    = eDate,
                                        frequency  = "W")
tmp_coy_RI[[length(tmp_coy_RI)+1]] = loopRI}

#This will remove all the lists that are NA.
#The model crashes if NA lists are included so they must be
removed.
tmp_coy_RI <- discard(tmp_coy_RI, ~ all(is.na(.x)))

#This transforms the list of lists into a single dataframe where
each list(firm) is a column.
coy_RI_data <- as.data.frame(tmp_coy_RI)

#Next the row names, which contain dates, is converted into a
date column.
#Step 1: take the row names and turn them into a column of
character type.
coy_RI_data$date <- rownames(coy_RI_data)
#Step 2: use the parse date time function, to turn characters
into POSIXct date format.
coy_RI_data$date <- parse_date_time(coy_RI_data$date, "ymd")
#This takes the dataset (dates as rows and firms as columns) and
turns it into list of firm/date combinations.
#It is also ordered in firm and date order.
coy_RI_data <- melt(coy_RI_data, id.vars = c("date"))
#This changes the column names of the dataset.
colnames(coy_RI_data) <- c("date","ISIN","coy_RI")

#Making the next stage of the object.
coy_data <- coy_RI_data
```

```
#Removal of objects that will be used by other functions later
on.
rm(i)

### The above process is now repeated for the other 4 periods (t-
1 to t-4, also referred to as tm1 to tm4).
### Commentary on the lines have been removed.
### Explanations of each step from the first iteration apply to
the matching lines in each other iteration.

##GETTING THE FIRM DATA FOR PERIOD t-1  ##

tmp_coy_RI_tm1 = list()

for (i in 1:length(coylist))
{ loopRItm1      <- dsws$timeSeriesRequest(instrument =
coylist[i],
                                           datatype   = "RI",
                                           startDate  = sDatetm1,
                                           endDate    = eDate,
                                           frequency  = "W")
tmp_coy_RI_tm1[[length(tmp_coy_RI_tm1)+1]] = loopRItm1}

tmp_coy_RI_tm1 <- discard(tmp_coy_RI_tm1, ~ all(is.na(.x)))
coy_RI_data_tm1 <- as.data.frame(tmp_coy_RI_tm1)
coy_RI_data_tm1$date <- rownames(coy_RI_data_tm1)
coy_RI_data_tm1$date <- parse_date_time(coy_RI_data_tm1$date,
"ymd")
coy_RI_data_tm1 <- melt(coy_RI_data_tm1, id.vars = c("date"))
colnames(coy_RI_data_tm1) <- c("date","ISIN","coy_RI")
coy_data_tm1 <- coy_RI_data_tm1
rm(i)

#GETTING THE FIRM DATA FOR PERIOD t-2 ##

tmp_coy_RI_tm2 = list()

for (i in 1:length(coylist))
{ loopRItm2      <- dsws$timeSeriesRequest(instrument =
coylist[i],
                                           datatype   = "RI",
                                           startDate  = sDatetm2,
                                           endDate    = eDate,
                                           frequency  = "W")
tmp_coy_RI_tm2[[length(tmp_coy_RI_tm2)+1]] = loopRItm2}

tmp_coy_RI_tm2 <- discard(tmp_coy_RI_tm2, ~ all(is.na(.x)))
coy_RI_data_tm2 <- as.data.frame(tmp_coy_RI_tm2)
coy_RI_data_tm2$date <- rownames(coy_RI_data_tm2)
coy_RI_data_tm2$date <- parse_date_time(coy_RI_data_tm2$date,
"ymd")
coy_RI_data_tm2 <- melt(coy_RI_data_tm2, id.vars = c("date"))
colnames(coy_RI_data_tm2) <- c("date","ISIN","coy_RI")
coy_data_tm2 <- coy_RI_data_tm2
rm(i)

#GETTING THE FIRM DATA FOR PERIOD t-3 ##

tmp_coy_RI_tm3 = list()

for (i in 1:length(coylist))
{ loopRItm3      <- dsws$timeSeriesRequest(instrument =
coylist[i],
                                           datatype   = "RI",
                                           startDate  = sDatetm3,
                                           endDate    = eDate,
                                           frequency  = "W")
tmp_coy_RI_tm3[[length(tmp_coy_RI_tm3)+1]] = loopRItm3}

tmp_coy_RI_tm3 <- discard(tmp_coy_RI_tm3, ~ all(is.na(.x)))
coy_RI_data_tm3 <- as.data.frame(tmp_coy_RI_tm3)
coy_RI_data_tm3$date <- rownames(coy_RI_data_tm3)
coy_RI_data_tm3$date <- parse_date_time(coy_RI_data_tm3$date,
"ymd")
coy_RI_data_tm3 <- melt(coy_RI_data_tm3, id.vars = c("date"))
colnames(coy_RI_data_tm3) <- c("date","ISIN","coy_RI")
coy_data_tm3 <- coy_RI_data_tm3
rm(i)

#GETTING THE FIRM DATA FOR PERIOD t-4 ##

tmp_coy_RI_tm4 = list()

for (i in 1:length(coylist))
{ loopRItm4      <- dsws$timeSeriesRequest(instrument =
coylist[i],
                                           datatype   = "RI",
                                           startDate  = sDatetm4,
                                           endDate    = eDate,
                                           frequency  = "W")
tmp_coy_RI_tm4[[length(tmp_coy_RI_tm4)+1]] = loopRItm4}

tmp_coy_RI_tm4 <- discard(tmp_coy_RI_tm4, ~ all(is.na(.x)))
coy_RI_data_tm4 <- as.data.frame(tmp_coy_RI_tm4)
coy_RI_data_tm4$date <- rownames(coy_RI_data_tm4)
coy_RI_data_tm4$date <- parse_date_time(coy_RI_data_tm4$date,
"ymd")
coy_RI_data_tm4 <- melt(coy_RI_data_tm4, id.vars = c("date"))
colnames(coy_RI_data_tm4) <- c("date","ISIN","coy_RI")
coy_data_tm4 <- coy_RI_data_tm4
```

```
rm(i)

## END OF GETTING THE FIRM RETURN DATA ##

#### ---- 06 - GETTING THE MARKET DATA -----
### The process of getting the weekly market return data is
repeated 5 times.
### This is repeated 5 times to ensure it returns data for each
possible week start and end combination,
### ie. Mon-Fri, Tue-Mon, Wed-Tue, Thu-Wed, Fri-Thu, referred to
as t-1 to t-4 (or tm1 to tm4) in the model.

#This pulls the list of markets from the static data as a vector
mktlist <- coy_s_data$INDX
#This removes duplicated values so it's a unique list of markets.
mktlist <- unique(mktlist)

##GETTING THE MARKET DATA FOR PERIOD t ##

#This creates a blank list that this iteration of weekly market
return data will be saved into.
tmp_mkt_RI = list()

#This loop takes each item on the firm list and then gets weekly
market return data from datastream.
#Results are saved as a list on the tmp_mkt_RI list.
for (i in 1:length(mktlist))
{ mkt_loop      <- dsws$timeSeriesRequest(instrument =
mktlist[i],
                                          datatype   = "RI",
                                          startDate  = sDate,
                                          endDate    = eDate,
                                          frequency  = "W")
tmp_mkt_RI[[length(tmp_mkt_RI)+1]] = mkt_loop}

#This will remove all the lists that are NA
#The model crashes if NA lists are included so they must be
removed.
tmp_mkt_RI <- discard(tmp_mkt_RI, ~ all(is.na(.x)))

#This transforms the list of lists into a single dataframe where
each list(mkt) is a column.
mkt_RI_data <- as.data.frame(tmp_mkt_RI)

#Next the row names, which contain dates, is converted into a
date column.
#Step 1: take the row names and turn them into a column of
character type.
mkt_RI_data$date <- rownames(mkt_RI_data)
#Step 2: use the parse date time function, to turn characters
```

```
into POSIXct date format.
mkt_RI_data$date <- parse_date_time(mkt_RI_data$date, "ymd")
#This takes the dataset (dates as rows and markets as columns)
and turns it into list of market/date combinations.
#It is also ordered in market and date order.
mkt_RI_data <- melt(mkt_RI_data, id.vars = c("date"))
#This changes the column names of the dataset.
colnames(mkt_RI_data) <- c("date","INDX","mkt_RI")

#Removal of objects that will be used by other functions later
on.
rm(i)

### The above process is now repeated for the other 4 periods (t-
1 to t-4, also referred to as tm1 to tm4).
### Commentary on the lines have been removed.
### Explanations of each step from the first iteration apply to
the matching lines in each other iteration.

##GETTING THE MARKET DATA FOR PERIOD t-1 ##

tmp_mkt_RI_tm1 = list()

for (i in 1:length(mktlist))
{ mkt_looptm1      <- dswsstimeSeriesRequest(instrument =
mktlist[i],
                                             datatype   = "RI",
                                             startDate  =
sDatetm1,
                                             endDate    = eDate,
                                             frequency  = "W")
tmp_mkt_RI_tm1[[length(tmp_mkt_RI_tm1)+1]] = mkt_looptm1}


tmp_mkt_RI_tm1 <- discard(tmp_mkt_RI_tm1, ~ all(is.na(.x)))
mkt_RI_data_tm1 <- as.data.frame(tmp_mkt_RI_tm1)
mkt_RI_data_tm1$date <- rownames(mkt_RI_data_tm1)
mkt_RI_data_tm1$date <- parse_date_time(mkt_RI_data_tm1$date,
"ymd")
mkt_RI_data_tm1 <- melt(mkt_RI_data_tm1, id.vars = c("date"))
colnames(mkt_RI_data_tm1) <- c("date","INDX","mkt_RI")
rm(i)

##GETTING THE MARKET DATA FOR PERIOD t-2##

tmp_mkt_RI_tm2 = list()

for (i in 1:length(mktlist))
{ mkt_looptm2      <- dsws$timeSeriesRequest(instrument =
mktlist[i],
                                             datatype   = "RI",
                                             startDate  =
sDatetm2,
                                             endDate    = eDate,
                                             frequency  = "W")
tmp_mkt_RI_tm2[[length(tmp_mkt_RI_tm2)+1]] = mkt_looptm2}

tmp_mkt_RI_tm2 <- discard(tmp_mkt_RI_tm2, ~ all(is.na(.x)))
mkt_RI_data_tm2 <- as.data.frame(tmp_mkt_RI_tm2)
mkt_RI_data_tm2$date <- rownames(mkt_RI_data_tm2)
mkt_RI_data_tm2$date <- parse_date_time(mkt_RI_data_tm2$date,
"ymd")
mkt_RI_data_tm2 <- melt(mkt_RI_data_tm2, id.vars = c("date"))
colnames(mkt_RI_data_tm2) <- c("date","INDX","mkt_RI")
rm(i)

##GETTING THE MARKET DATA FOR PERIOD t-3##

tmp_mkt_RI_tm3 = list()

for (i in 1:length(mktlist))
{ mkt_looptm3      <- dsws$timeSeriesRequest(instrument =
mktlist[i],
                                             datatype   = "RI",
                                             startDate  =
sDatetm3,
                                             endDate    = eDate,
                                             frequency  = "W")
tmp_mkt_RI_tm3[[length(tmp_mkt_RI_tm3)+1]] = mkt_looptm3}

tmp_mkt_RI_tm3 <- discard(tmp_mkt_RI_tm3, ~ all(is.na(.x)))
mkt_RI_data_tm3 <- as.data.frame(tmp_mkt_RI_tm3)
mkt_RI_data_tm3$date <- rownames(mkt_RI_data_tm3)
mkt_RI_data_tm3$date <- parse_date_time(mkt_RI_data_tm3$date,
"ymd")
mkt_RI_data_tm3 <- melt(mkt_RI_data_tm3, id.vars = c("date"))
colnames(mkt_RI_data_tm3) <- c("date","INDX","mkt_RI")
rm(i)

##GETTING THE MARKET DATA FOR PERIOD t-4##

tmp_mkt_RI_tm4 = list()

for (i in 1:length(mktlist))
{ mkt_looptm4      <- dsws$timeSeriesRequest(instrument =
mktlist[i],
                                             datatype   = "RI",
                                             startDate  =
sDatetm4,
                                             endDate    = eDate,
                                             frequency  = "W")
tmp_mkt_RI_tm4[[length(tmp_mkt_RI_tm4)+1]] = mkt_looptm4}

tmp_mkt_RI_tm4 <- discard(tmp_mkt_RI_tm4, ~ all(is.na(.x)))
mkt_RI_data_tm4 <- as.data.frame(tmp_mkt_RI_tm4)
mkt_RI_data_tm4$date <- rownames(mkt_RI_data_tm4)
mkt_RI_data_tm4$date <- parse_date_time(mkt_RI_data_tm4$date,
"ymd")
mkt_RI_data_tm4 <- melt(mkt_RI_data_tm4, id.vars = c("date"))
colnames(mkt_RI_data_tm4) <- c("date","INDX","mkt_RI")
rm(i)

## END OF GETTING THE MARKET RETURN DATA ##

#### ---- 07 - MERGING FIRM, MARKET AND STATIC DATA ----

### MERGING PERIOD t

#This merges the static and time-dependent firm data on the basis
of the firm's ISIN value.
coy_data_merged <- merge(coy_data, coy_s_data, by = "ISIN", all.y
= TRUE)

#This merges the firm data and the market data.
data <- merge(coy_data_merged, mkt_RI_data, by = c("date",
"INDX"))

#This adds an identifier of the firm ISIN, year and week number
of the year
data <- data %>% mutate(id = paste(ISIN, year(date),week =
week(date)))
# end of merging coy and static data for period t

### The above process is now repeated for the other 4 periods (t-
1 to t-4, also referred to as tm1 to tm4).
### Commentary on the lines have been removed.

### MERGING PERIOD t-1

coy_data_merged_tm1 <- merge(coy_data_tm1, coy_s_data, by =
"ISIN", all.y = TRUE)
data_tm1 <- merge(coy_data_merged_tm1, mkt_RI_data_tm1, by =
c("date", "INDX"))
data_tm1 <- data_tm1 %>% mutate(id = paste(ISIN, year(date),week
= week(date)))

### MERGING PERIOD t-2

coy_data_merged_tm2 <- merge(coy_data_tm2, coy_s_data, by =
"ISIN", all.y = TRUE)
```

```
data_tm2 <- merge(coy_data_merged_tm2, mkt_RI_data_tm2, by =
c("date", "INDX"))
data_tm2 <- data_tm2 %>% mutate(id = paste(ISIN, year(date),week
= week(date)))

### MERGING PERIOD t-3

coy_data_merged_tm3 <- merge(coy_data_tm3, coy_s_data, by =
"ISIN", all.y = TRUE)
data_tm3 <- merge(coy_data_merged_tm3, mkt_RI_data_tm3, by =
c("date", "INDX"))
data_tm3 <- data_tm3 %>% mutate(id = paste(ISIN, year(date),week
= week(date)))

### MERGING PERIOD t-4

coy_data_merged_tm4 <- merge(coy_data_tm4, coy_s_data, by =
"ISIN", all.y = TRUE)
data_tm4 <- merge(coy_data_merged_tm4, mkt_RI_data_tm4, by =
c("date", "INDX"))
data_tm4 <- data_tm4 %>% mutate(id = paste(ISIN, year(date),week
= week(date)))

## END OF MERGING FIRM, MARKET AND STATIC DATA ##

#### ---- 08 - CALCULATING THE OBSERVED RETURNS ----

## PERIOD t

#This calculates new variables of lagged firm and market returns
#NB ensure dplyr is enabled otherwise it will calculate incorrect
values (but won't give error message).
data   <- data %>% group_by(NAME) %>%
mutate(CoyLagRI=log(coy_RI)-log(lag(coy_RI)))
data   <- data %>% group_by(NAME) %>%
mutate(MktLagRI=log(mkt_RI)-log(lag(mkt_RI)))
#This removes the rows where there are missing values in lagged
firm and market returns
data <- data[complete.cases(data[ , c("CoyLagRI","MktLagRI")]),]
#This removes firms that have less than minimum number of weeks
(as per Stocks Import Template Excel file)
data <- data %>% group_by(NAME) %>% filter(n() > weeks)

### The above process is now repeated for the other 4 periods (t-
1 to t-4, also referred to as tm1 to tm4).
### Commentary on the lines have been removed.

## PERIOD t-1

data_tm1   <- data_tm1 %>% group_by(NAME) %>%
mutate(CoyLagRI=log(coy_RI)-log(lag(coy_RI)))
data_tm1   <- data_tm1 %>% group_by(NAME) %>%
mutate(MktLagRI=log(mkt_RI)-log(lag(mkt_RI)))
data_tm1 <- data_tm1[complete.cases(data_tm1[ ,
c("CoyLagRI","MktLagRI")]),]
data_tm1 <- data_tm1 %>% group_by(NAME) %>% filter(n() >weeks)

## PERIOD t-2

data_tm2   <- data_tm2 %>% group_by(NAME) %>%
mutate(CoyLagRI=log(coy_RI)-log(lag(coy_RI)))
data_tm2   <- data_tm2 %>% group_by(NAME) %>%
mutate(MktLagRI=log(mkt_RI)-log(lag(mkt_RI)))
data_tm2 <- data_tm2[complete.cases(data_tm2[ ,
c("CoyLagRI","MktLagRI")]),]
data_tm2 <- data_tm2 %>% group_by(NAME) %>% filter(n() >weeks)

## PERIOD t-3

data_tm3   <- data_tm3 %>% group_by(NAME) %>%
mutate(CoyLagRI=log(coy_RI)-log(lag(coy_RI)))
data_tm3   <- data_tm3 %>% group_by(NAME) %>%
mutate(MktLagRI=log(mkt_RI)-log(lag(mkt_RI)))
data_tm3 <- data_tm3[complete.cases(data_tm3[ ,
c("CoyLagRI","MktLagRI")]),]
data_tm3 <- data_tm3 %>% group_by(NAME) %>% filter(n() >weeks)

## PERIOD t-4

data_tm4   <- data_tm4 %>% group_by(NAME) %>%
mutate(CoyLagRI=log(coy_RI)-log(lag(coy_RI)))
data_tm4   <- data_tm4 %>% group_by(NAME) %>%
mutate(MktLagRI=log(mkt_RI)-log(lag(mkt_RI)))
data_tm4 <- data_tm4[complete.cases(data_tm4[ ,
c("CoyLagRI","MktLagRI")]),]
data_tm4 <- data_tm4 %>% group_by(NAME) %>% filter(n() >weeks)

## END OF CALCULATING THE OBSERVED RETURNS ##

#### ---- 09 - CALCULATING THE AMIHUD ADJUSTMENT ----

#First step is to collect additional data required - it is
similar to what was done in section 05
#This loop takes each item on the firm list and then gets daily
firm return data from datastream.
#There is no need to repeat this 5 times because it is daily
data.
#This same amihud measure will apply in data t to tm4.

tmp_coy_RI = list()
```

```
for (i in 1:length(coylist))
{ loopRId      <- dsws$timeSeriesRequest(instrument = coylist[i],
                                         datatype   = "RI",
                                         startDate  = sDate,
                                         endDate    = eDate,
                                         frequency  = "D")
tmp_coy_RI[[length(tmp_coy_RI)+1]] = loopRId}

tmp_coy_RI <- discard(tmp_coy_RI, ~ all(is.na(.x)))
coy_RI_datad <- as.data.frame(tmp_coy_RI)
coy_RI_datad$date <- rownames(coy_RI_datad)
coy_RI_datad$date <- parse_date_time(coy_RI_datad$date, "ymd")
coy_RI_datad <- melt(coy_RI_datad, id.vars = c("date"))
colnames(coy_RI_datad) <- c("date","ISIN","coy_RI")
rm(i)

#Firm daily volume data - this is the same process as above, only
for volumes instead of returns.
tmp_coy_VA = list()
for (i in 1:length(coylist))
{ loopVAd      <- dsws$timeSeriesRequest(instrument = coylist[i],
                                         datatype   = "VA",
                                         startDate  = sDate,
                                         endDate    = eDate,
                                         frequency  = "D")
tmp_coy_VA[[length(tmp_coy_VA)+1]] = loopVAd}

tmp_coy_VA <- discard(tmp_coy_VA, ~ all(is.na(.x)))
coy_VA_datad <- as.data.frame(tmp_coy_VA)
coy_VA_datad$date <- rownames(coy_VA_datad)
coy_VA_datad$date <- parse_date_time(coy_VA_datad$date, "ymd")
coy_VA_datad <- melt(coy_VA_datad, id.vars = c("date"))
colnames(coy_VA_datad) <- c("date","ISIN","coy_VA")
rm(i)

#This merges the firm daily return and volume data into a single
dataframe.
coy_data_d <- merge(coy_RI_datad, coy_VA_datad, by = c("ISIN",
"date"), all.x = TRUE)

#Calculating the amihud value:

#Calculate lagged firm return value.
coy_data_d <- coy_data_d %>% group_by(ISIN) %>%
mutate(CoyLagRI=log(coy_RI)-log(lag(coy_RI)))
#Remove rows with missing values in returns, volumes and lagged
daily returns .
coy_data_d <- coy_data_d[complete.cases(coy_data_d[ ,
c("coy_RI","coy_VA","CoyLagRI")]),]
#Merging the daily data with the static firm info.
```

```
coy_data_d <- merge(coy_data_d, coy_s_data, by = "ISIN")
#This adds an identifier of the firm ISIN, year and week number
of the year.
coy_data_d <- coy_data_d %>% mutate(id = paste(ISIN,
year(date),week = week(date)))
#This creates a boolean variable that contains TRUE or FALSE if
there was trading on that day.
coy_data_d <- coy_data_d %>% mutate(traded =
ifelse(is.na(coy_VA), FALSE, ifelse(coy_VA == 0, FALSE, TRUE)))
#This creates a new dataframe that contains the number of days
traded for each week and for each firm.
days_traded <- coy_data_d %>% group_by(id) %>% summarise(days =
sum(traded))
#This adds a new variable that contains TRUE or FALSE based on
whether there enough traded days in this week (as per Stocks
Import Template Excel file).
days_traded <- days_traded %>% mutate(vol_incl = days >=
min_days_traded)
#This created the absolute value of the return
coy_data_d <- coy_data_d %>% mutate(abs_ret = abs(CoyLagRI))
#This created measure dy, which is the amihud measure for that
day
coy_data_d <- coy_data_d %>% mutate(measure_dy = abs_ret/coy_VA)
#Sigma measure is the amihud measure for that firm for the week,
which is the sum of the daily measures.
sigma_measure <- coy_data_d %>% group_by(id) %>%
summarise(measure_m = sum(measure_dy), na.rm=T)
#This joins the sigma measure to the the number of days traded
amihud <- sigma_measure %>% left_join(days_traded, by="id") %>%
filter(vol_incl==TRUE)
#this calculates the amihud measure that we use for our screening
process.
amihud <- amihud %>% mutate(amihud = (1/days)*measure_m*1000000)
#joining the amihad measure back to the firm daily returns
coy_data_d <- left_join(amihud, coy_data_d, by = "id")
#This makes a new dataframe which contains the weekly average
amihud measure
amihud_weekly <- coy_data_d %>% group_by(id) %>%
summarise(mean_amihud_week = mean(amihud))
#This dataframe has one line for each firm week and the average
amihud value for that firm week.

## END OF CALCULATING THE AMIHUD ADJUSTMENT ##

#### ---- 10 - MERGING THE WEEKLY DATA WITH THE AMIHUD MEASURE --
--

## PERIOD t
#This creates a data frame combining all the firm and market data
with the amihud number for each firm week.
```

```
data <- merge(data, amihud, by = "id")
#This filters the dataset to remove all the firm weeks with an
amihud greater than the cutoff specified in the template.
data <- data[data[,"amihud"] < amihud_cutoff,]
#This removes all the firms that have less than the minimum
number of weeks of observations as specified in the template.
data <- data %>% group_by(ISIN) %>% filter(n() > weeks)

### The above process is now repeated for the other 4 periods (t-
1 to t-4, also referred to as tm1 to tm4).
### Commentary on the lines have been removed.

## PERIOD t-1
data_tm1 <- merge(data_tm1, amihud, by = "id")
data_tm1 <- data_tm1[data_tm1[,"amihud"] < amihud_cutoff,]
data_tm1 <- data_tm1 %>% group_by(ISIN) %>% filter(n() > weeks)

## PERIOD t-2
data_tm2 <- merge(data_tm2, amihud, by = "id")
data_tm2 <- data_tm2[data_tm2[,"amihud"] < amihud_cutoff,]
data_tm2 <- data_tm2 %>% group_by(ISIN) %>% filter(n() > weeks)

## PERIOD t-3
data_tm3 <- merge(data_tm3, amihud, by = "id")
data_tm3 <- data_tm3[data_tm3[,"amihud"] < amihud_cutoff,]
data_tm3 <- data_tm3 %>% group_by(ISIN) %>% filter(n() > weeks)

## PERIOD t-4
data_tm4 <- merge(data_tm4, amihud, by = "id")
data_tm4 <- data_tm4[data_tm4[,"amihud"] < amihud_cutoff,]
data_tm4 <- data_tm4 %>% group_by(ISIN) %>% filter(n() > weeks)

#This step creates the combined dataset with all the weekly
returns for all 5 periods.
data_all <- rbind(data, data_tm1, data_tm2, data_tm3, data_tm4)

## END OF MERGING THE WEEKLY DATA WITH THE AMIHUD MEASURE ##

#### ---- 11 - REGRESSING THE RETURNS ----

#This clears out the NA's for lagged firm and market returns.
data_all <- data_all[complete.cases(data_all[ ,
c("CoyLagRI","MktLagRI")]),]
#This does the regression modelling fitting a linear model for
each firm.
params_all <- data_all %>% group_by(NAME) %>% do(tidy(lm(CoyLagRI
~ MktLagRI, data = .)))
## END OF REGRESSING THE RETURNS ##

#### ---- 12 - VASICEK ADJUSTMENT ----
```

```
# Calculate sample variance of calculated betas.
params_all <- params_all %>% mutate(std.var = std.error^2)
var_est_betas <-
var(params_all$estimate[which(params_all$term=="MktLagRI")])
# Calculate lambda parameter.
params_all <- params_all %>% mutate(lambda =
var_est_betas/(var_est_betas + std.var))
# Calculate Vasicek adjustment: Bv=Bv_hat*(1-lambda)+lambda*B_ols
(where Bv_hat = 1).
avg_beta <-
mean(params_all$estimate[which(params_all$term=="MktLagRI")])
params_all <- params_all %>% mutate(vas = (avg_beta*(1-
lambda))+(lambda*estimate))

## END OF VASICEK ADJUSTMENT ##

#### ---- 13 - GETTING THE GEARING DATA ----

#Gearing is annual and thus does not need loop be repeated 5
times as per weekly data.
#This runs the data loop to get the debt data - WC03255 = Total
debt.
tmp_coy_debt = list()

for (i in 1:length(coylist))
{ loopdebt      <- dsws$timeSeriesRequest(instrument =
coylist[i],
                                          datatype   = 'WC03255',
                                          startDate  = sDate,
                                          endDate    = eDate,
                                          frequency  = "Y")
tmp_coy_debt[[length(tmp_coy_debt)+1]] = loopdebt}

#This will remove all the lists that are NA.
tmp_coy_debt <- discard(tmp_coy_debt, ~ all(is.na(.x)))

coy_debt_data <- as.data.frame(tmp_coy_debt)
coy_debt_data$date <- rownames(coy_debt_data)
coy_debt_data$date <- parse_date_time(coy_debt_data$date, "ymd")
coy_debt_data <- melt(coy_debt_data, id.vars = c("date"))
colnames(coy_debt_data) <- c("date","ISIN","totaldebt")
#renaming the object
totaldebt <- coy_debt_data
rm(i)

#This runs the loop for the market cap data - WC08001 = market
capitalization.
tmp_coy_mktcap = list()
for (i in 1:length(coylist))
{ loopmc      <- dsws$timeSeriesRequest(instrument = coylist[i],
                                        datatype   = 'WC08001',
                                        startDate  = sDate,
                                        endDate    = eDate,
                                        frequency  = "Y")
tmp_coy_mktcap[[length(tmp_coy_mktcap)+1]] = loopmc}

#This will remove all the lists that are NA.
tmp_coy_mktcap <- discard(tmp_coy_mktcap, ~ all(is.na(.x)))

coy_mktcap_data <- as.data.frame(tmp_coy_mktcap)
coy_mktcap_data$date <- rownames(coy_mktcap_data)
coy_mktcap_data$date <- parse_date_time(coy_mktcap_data$date,
"ymd")
coy_mktcap_data <- melt(coy_mktcap_data, id.vars = c("date"))
colnames(coy_mktcap_data) <- c("date","ISIN","mktcap")
#renaming the object
marketcap <- coy_mktcap_data
rm(i)

## END OF GETTING THE GEARING DATA ##

#### ---- 14 - CALCULATING THE GEARING RATIOs ----
#Now join debt and market cap data to calculate gearing
gearingdata <- left_join(totaldebt, marketcap, by =
c("ISIN","date"))
#Now calculate gearing variable: total value of the firm (ie.
total debt + total equity)
gearingdata <- gearingdata %>% mutate(totalcap = totaldebt +
mktcap)
#This calculates gearing percentage (ie. the proportion of the
total firm value that is debt)
gearingdata <- gearingdata %>% mutate(firm_DE = totaldebt /
mktcap)
#Removing all the firms where gearing ratio is NA due to missing
data
gearingdata <- gearingdata[complete.cases(gearingdata[ ,
c("firm_DE")]),]

#Now clean the data and aggregate to calculate average gearing
ratio
#This line clears out all the NA's
gearingdata_clean <- gearingdata[complete.cases(gearingdata[ ,
c("firm_DE")]),]
#This calculates the average gearing ratio for each firm in the
sample.
gearingdata_ag <- aggregate(x = gearingdata_clean[,"firm_DE"], by
= list(gearingdata_clean$ISIN), FUN = mean)
#This changes the column names to enable easier joining with
other dataframes
colnames(gearingdata_ag) <- c("ISIN", "firm_DE_ratio")
```

```
## END OF CALCULATING THE GEARING RATIOs ##

#### ---- 15 - JOINING THE PARAMS AND GEARING DATA ----

#This joins static data to parameters by "NAME" first, as params
has no ISIN numbers, but gearing data only has ISIN code.
params_all <- left_join(params_all, coy_s_data, by = "NAME")
#This joins the parameters and the gearing data.
params_all <- left_join(params_all, gearingdata_ag, by = "ISIN")
#This calculates the unlevered beta .
params_all <- params_all %>% mutate(unlevered_beta = vas / (1 +
firm_DE_ratio))
#This relevers the beta with the targeted gearing ratio.
params_all <- params_all %>% mutate(relevered_beta =
unlevered_beta * (1 + target_DE))

## END OF JOINING THE PARAMS AND GEARING DATA ##

#### ---- 16 - FINESSING THE FINAL OUTPOUTS FOR EXPORT ----

#The combined object is renamed as final output
final_output <- params_all

# This creates object "fo_trim" - a trimmed and clean version of
the final outputs.
#First remove rows containing intercepts.
fo_trim <- final_output[final_output[,"term"]!="(Intercept)",]
#This removes the NA's.
fo_trim <- fo_trim[complete.cases(fo_trim[ ,
c("estimate","relevered_beta")]),]
#This renames some column headings to make interpretation clearer
fo_trim <- fo_trim %>% rename(ols_beta = estimate, vasicek_beta =
vas)
#This removes some of the columns that are not needed
fo_trim[c("BETA","MNEM", "ISINID")] <- NULL

## This creates a version of the gearing data for export into
excel for the outputs package
gearingdata_exp <- aggregate(x =
gearingdata_clean[,c("totaldebt", "mktcap", "totalcap",
"firm_DE")], by = list(gearingdata_clean$ISIN), FUN = mean)

#This outputs parameter file, parameter file excluding
intercepts, and gearing data to spreadsheets.
write.xlsx(final_output, file = paste0("final_output_", filename,
".xlsx"))
write.xlsx(fo_trim, file = paste0("final_output_trimmed_",
filename, ".xlsx"))
write.xlsx(gearingdata_exp, file =
paste0("gearing_data_cleaned_", filename, ".xlsx"))
```

```
#This saves the R environment
save.image(file = paste0("R_objects_", filename, ".RData"))

## END OF FINESSING THE FINAL OUTPOUTS FOR EXPORT ##

#### ---- 17 - OUTPUT CHARTS ----

### First create the colours and IPART theme objects for charts.
# Colours
iptblu <- rgb(0,123,196, maxColorValue = 255); iptgre <-
rgb(160,160,154, maxColorValue = 255)
iptoj <- rgb(247,141,30, maxColorValue = 255)
# Define Ipart theme.
theme_ipart <- theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5),
          axis.title.y = element_text(margin = margin(t=0, r=20,
b=0, l=0), size=16, colour="#707070"),
          axis.title.x = element_text(margin = margin(t=20, r=20,
b=0, l=0), size=16,colour="#707070"),
          axis.text.x = element_text(size=16,colour="#707070"),
          axis.text.y = element_text(size=16, colour="#707070"),
          strip.text.x = element_text(size = 16, colour =
"#707070"),
          legend.position = "bottom",
          legend.text = element_text(size=16, margin =
margin(t=10, r=5, b=0, l=0)),
          legend.title = element_blank())
### End of colour and theme creation.

### Data needs to be manipulated before being charted.
#Creating the new object that charts will draw from.
betas <- params_all

#Renaming the columns of the new object to ensure it matches
previous charts.
colnames(betas) <- c("name", "term", "OLS_Beta", "std.error",
"staticitc", "p.value", "std.var", "lambda",
                     "Vas_Beta", "Symbol", "ISIN", "BETA",
"ESTAT", "INDX", "MNEM", "ISINID",
                     "firm_DE_ratio", "Unlev_Beta", "Relev_Beta")

#This removes the intercept rows.
betas <- betas[betas[,"term"]!="(Intercept)",]

#This creates the group medians of each variable for the charts.
median_beta_unlev <- round(median(betas$Unlev_Beta, na.rm=T),3)
median_beta_relev <- round(median(betas$Relev_Beta, na.rm=T),3)
median_vas_beta   <- round(median(betas$Vas_Beta),3)
median_beta       <- round(median(betas$OLS_Beta),3)
```

```
### Here we create the two charts.
## Dotplot1: Unlevered -> Relevered
(dotplot1 <- ggplot(betas, aes(x= Relev_Beta,
y=fct_reorder(Symbol, Relev_Beta))) +
        geom_point(aes(x= Unlev_Beta, y=fct_reorder(Symbol,
Relev_Beta)), col = iptblu) +
        geom_point(aes(x= Relev_Beta, y=fct_reorder(Symbol,
Relev_Beta)), col = "red") +
        geom_segment(aes(x=betas$Unlev_Beta, y=Symbol,
xend=betas$Relev_Beta, yend=Symbol), size=0.3, col=iptblu) +
        geom_vline(xintercept = 1, col='grey')+
        geom_vline(xintercept = median_beta_relev, col='red') +
        #geom_vline(xintercept = beta_relev, col='green') +
        annotate('text',x=
median_beta_relev,y=1.5,label=paste0('Median Relevered beta =
',median_beta_relev), col='red',size=6) +
        #geom_text_repel()
        theme_ipart +
        labs(x= 'Relevered beta after Vasicek adjustment',
             y= 'Company code'))

ggsave(dotplot1,
file=paste0('unlev_to_relev_dotplot_2_',filename,'.png'), width =
20, height = 10)

## Dotplot2 - OLS -> Vasicek adjustment
(dotplot2 <- ggplot(betas, aes(x= Vas_Beta, y=fct_reorder(Symbol,
Vas_Beta))) +
        geom_point(aes(x= OLS_Beta, y=fct_reorder(Symbol,
Vas_Beta)), col = iptblu) +
        geom_point(aes(x= Vas_Beta, y=fct_reorder(Symbol,
Vas_Beta)), col = "red") +
        geom_segment(aes(x=betas$OLS_Beta, y=Symbol,
xend=betas$Vas_Beta, yend=Symbol), size=0.3, col=iptblu) +
        geom_vline(xintercept = 1, col='grey')+
        geom_vline(xintercept = median_beta, col='red') +
        #geom_vline(xintercept = beta_relev, col='green') +
        annotate('text',x=
median_vas_beta,y=1.5,label=paste0('Median Vasicek beta =
',median_vas_beta), col='red',size=6) +
        #geom_text_repel()
        theme_ipart +
        labs(x= 'Beta after Vasicek adjustment',
             y= 'Company code'))

ggsave(dotplot2,
file=paste0('OLS_to_vas_dotplot_2_',filename,'.png'), width=20,
height=10)

## END OF OUTPUT CHARTS ##
```

```
#### ---- END OF MODEL ----
```