```
#### IPART Beta Estimation Model
## Version 1.19
## This version date: 4/02/2019
## Author: IPART

##############################    1. Set libraries, inputs       ##################################################################
# Clear workspace
rm(list=ls())
# Libraries
library(tidyverse); library(RDatastream); library(readxl); library(broom); library(tictoc); library(lubridate)
library(ggrepel); library(plotly); library(openxlsx); library(rlist)
#NB: RDatastream is not available on CRAN. To access, use: devtools::install_github("fcocquemas/RDataStream")
# Colours
iptblu <- rgb(0,123,196, maxColorValue = 255); iptgre <- rgb(160,160,154, maxColorValue = 255)
iptoj <- rgb(246,139,31, 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())

## Inputs ---------------------------------------------------------------------------------------------------------
# Select working directory
```

```
winDialog(type='ok','Please select your working directory for this analysis')
wd <- choose.dir(default='Your_directory_path')
if(is.na(wd)){stop("Need a valid working directory to run model.")}
winDialog(type='ok','Please select your input template')   #Select template input file
file <- choose.files()

# Extract relevant inputs
inp          <- read_excel(file,sheet='Input')
orig_sample   <- inp$Symbol
n0           <- data_frame(firms=length(orig_sample), filter = "Original sample after market filter") #Use data.frame() or tibble() here - data_frame is deprecated
sDate         <- as.Date(inp$`Start Date`[1])
eDate         <- as.Date(inp$`End Date`[1])
amihud_cutoff <- as.numeric(inp$`Amihud Cutoff`[1])
target_de     <- as.numeric(inp$`Target DE`[1])

##Set datastream access
user<-list(username='Your_datastream_username', password='Your_datastream_password')

################################    2. Background work: Static, MKT, and Rf data     ###################################################
## Can apply additional industry filter if necessary
#industry <- ds(user,requests = paste0(orig_sample, "~XREF"))
#industry <- lapply(industry['Data',], data.frame)
#industry <- list.stack(industry)

##Get static identifier data for sample
static <- ds(user,requests = paste0(orig_sample,"~=NAME,ISIN,BETA,INDX~REP"))

info <- lapply(static['Data',], data.frame)
info <- list.stack(info) %>% tbl_df()
```

```
info <- map_dfr(static['Data',],data.frame)
info <- info %>% mutate_at(vars(-DATE), .funs = as.character)   # converts factors to strings
info <- info %>% mutate(BETA = as.numeric(BETA))
names(info) <- c('ds_beta', 'ccy','date','indx','isin','name','symbol')

info <- map_dfr(static['Data',],data.frame)%>% #map_dfr (purrr) will coerce factors to characters in the bind
  mutate(BETA=as.numeric(BETA))
names(info) <- c('ds_beta', 'ccy','date','indx','isin','name','symbol')

# Error collection
no_isin <-info %>% filter(isin=='NA') %>% select(symbol)    # remove NA (characters) for ISIN
no_indx <-info %>% filter(indx=='NA') %>% select(symbol)    # remove NA (characters) for INDX

#Clean static info
(info <- info %>% filter(!isin=='NA',!indx=='NA'))          # removed here to save uneccessarily pulling rf, rm data

sample_coys <- info$symbol                                  # new sample
sample_isin <- info$isin                                    # used to generate bloomberg tickers


# 2.1  Get relevant MKT and rf data based on unique markets
# (source data needs to be updated occasionally)
setwd('Your_directory_path')
load('mkt_and_rf.RData')                                    #load in full df's of rm, rmx, and rf
mkt_universe <- read.csv(file='mkt_list.csv', header=T, stringsAsFactors = F)
setwd(wd)                                                   # revert back to selected wd

# Add relevant series
info <- info %>% left_join(mkt_universe, by= c('indx' = 'mkt'))    # pull in relevant Rf series, if country doesn't appear on list - will return NA
```

```
info <- info %>% mutate(tax_rate = as.numeric(tax_rate))

rf   <- filter(df_rf,Rf %in% unique(info$Rf))                  # produces a tidy df of monthly risk free returns for each country
rm   <- filter(df_rm, mkt %in% unique(info$indx))                 # "" market returns ""
rmx  <- filter(df_rmx, mkt %in% unique(info$indx))                 # "" excess returns ""
rm(df_rf,df_rm,df_rmx)


########################################     3. Access data      ###############################################################################
## 3.1 Get raw data (or load pre-saved data)
answer <- winDialog(type = 'yesno', 'Would you like to load previously saved data?')
if(answer == 'YES'){
  winDialog(type= 'ok', 'Select data_d')
  pre_saved_daily <- choose.files(default='Your_directory_path')
  load(pre_saved_daily)
  winDialog(type= 'ok', 'Select data_m')
  pre_saved_monthly <- choose.files(default='Your_directory_path ')
  load(pre_saved_monthly)
} else {
  print('Accessing Datastream - Takes around 15 mins for every 100 firms')
  # daily data (can take up to 30 mins)
  tic()
  data_d<-ds(user,sample_coys,c('RI','VA'),fromDate=sDate,toDate=eDate,period='D')  # Daily data for Amihud liquidity measure
  toc()
  save(data_d, file='data_d.RData')                                    # Save to wd() for offline use
  # monthly data
  data_m<-ds(user,sample_coys,c('RI','VA'),fromDate=sDate,toDate=eDate,period='M')  # Monthly data for estimation
  save(data_m, file='data_m.RData')
}
```

```
# Note that the connection to DS can sometimes fail resulting in an error: "Conntection was reset"
# This is more likely for large queries (>200 companies)
# Re-running the code generally fixes the problem

## Cleaning ----------------------------------------------------------------------------------------------------------
# 3.2  Condense into single data frame
daily   <- lapply(data_d['Data',], data.frame)                          # Convert to list of data frames
daily   <- list.stack(daily, fill=TRUE) %>% tbl_df()                      # Stack
if(length(names(daily)) == 8){
  names(daily) <- c('ccy','date','name','freq','Ri_val','symbol','turnover','error')
  daily <- daily %>% select(date, symbol, Ri_val, turnover)
} else{
  names(daily) <- c('ccy','date','name','freq','Ri_val','symbol','turnover')  # Rename (DS inserts 'INSTERROR' in name if error)
  daily <- daily %>% select(date, symbol, Ri_val, turnover)                 # Select
}
# 3.3  Find connection failures
connect_fails <- lapply(data_d['StatusType',],data.frame)           # Looks in the ds return (status) for failures
connect_fails <- list.stack(connect_fails) %>% mutate(symbol = sample_coys)
connect_fails <- connect_fails %>% filter(X..i..=='Failure') %>% select(symbol) %>% tbl_df() #As above on tbl_df()

# 3.4  Find dead / suspended
tmp   <- str_detect(daily$Ri_val, "FAILED") # ds doesn't pull data for coys that have stopped trading (need to remove from sample)
if(sum(tmp, na.rm=T) == 0){
  print("All sample companies are active")
  dead <- tibble()   # empty data frame
} else {
  dead  <- daily[tmp,] %>% select(turnover) %>%             # symbol for error companies appears in the turnover column
    distinct() %>% rename(symbol=turnover) %>% na.omit()    # unique symbols
}
```

```
# 3.5  Collect errors
errors <- list('Dead'= dead, 'Connect failures'= connect_fails, 'No ISIN'=no_isin, 'No INDX'=no_indx)
(error_list  <- bind_rows(errors, .id = 'Error_Type'))
# Remove from our sample coys & data
keep    <- !sample_coys %in% error_list$symbol
sample1 <- sample_coys[keep]

## 3.6  DAILY
daily  <- daily %>% filter(symbol %in% sample1)   # keep non-error firms
n_distinct(daily$symbol) == length(sample1)       # check
n1 <- data_frame(firms = n_distinct(daily$symbol), filter = "Data quality: Connect failures, dead, no ISIN, no INDX")

# 3.7  Data quality tests
# Test 1 - Find NAs in symbol col
if(any(is.na(daily$symbol))==TRUE){
  print("NAs present in the symbol column - needs fixing")
}else{print("Test 1: Success")}
# Test 2 - "Failed"
if(sum(str_detect(daily$Ri_val, "FAILED"), na.rm=T) > 0){
  print("Error terms still present")
}else{print("Test 2: Success")}
# Coerce vars to correct format
daily <- daily %>% mutate_at(vars(Ri_val,turnover), .funs = as.character)
daily <- daily %>% mutate_at(vars(Ri_val,turnover), .funs = as.numeric)
daily <- daily %>% mutate(symbol = as.character(symbol))

##  3.8  MONTHLY
monthly <- lapply(data_m['Data',], data.frame)
```

```
monthly <- list.stack(monthly, fill=TRUE) %>% tbl_df()
if(length(names(monthly)) == 8){
  names(monthly) <- c('ccy','date','name','freq','Ri_val','symbol','turnover','error')
  monthly <- monthly %>% select(date, symbol, Ri_val, turnover)
} else{
  names(monthly) <- c('ccy','date','name','freq','Ri_val','symbol','turnover')  # Rename (DS inserts 'INSTERROR' in name if error)
  monthly <- monthly %>% select(date, symbol, Ri_val, turnover)
}

monthly  <- monthly %>% filter(symbol %in% sample1)      # keep non-error firms
# Find NAs in symbol col
if(any(is.na(monthly$symbol))==TRUE){
  print("NA present in the symbol column")
}
# Test again for "Failed"
if(sum(str_detect(monthly$Ri_val, "FAILED"), na.rm=T) > 0){
  print("Error terms still present")
}

# 3.9  Coerce vars
monthly <- monthly %>% mutate_at(vars(Ri_val,turnover), .funs = as.character)
monthly <- monthly %>% mutate_at(vars(Ri_val,turnover), .funs = as.numeric)
monthly <- monthly %>% mutate(symbol = as.character(symbol))

# 3.10 Calculate log returns ----------------------------------------------------------------------------------------------
# 3.10.1  For daily and monthly prices
daily   <- daily %>% group_by(symbol) %>% mutate(Ri=log(Ri_val)-log(lag(Ri_val)))     # log daily return
monthly <- monthly %>% group_by(symbol) %>% mutate(Ri=log(Ri_val)-log(lag(Ri_val)))
```

```
## 3.10.2  Calculate excess (monthly) returns
# monthly returns only (daily rets only used in Amihud calculation)
rftmp   <- info %>% select(symbol, Rf)
monthly <- monthly %>% left_join(rftmp, by='symbol') %>%                    # add rf code for each country then add rf return data
  left_join(rf, by=c('date', 'Rf'))                                         # join rf code + actual rf for each month
monthly <- monthly %>% mutate(id = paste(symbol,year(date),mth = month(date)))  # combines 2 tmp vars to form id
monthly <- monthly %>% mutate(Rx = Ri - return)                             # calc excess returns (Ri - Rf)
rm(rftmp)
monthly <- monthly %>% rename(Rf_code = Rf, Rf = return)                    # Tidy names


#################################     4. Calculate monthly Amihud measure    ###################################################
# 4.1 create ID + find monthly trade volume
daily <- daily %>% mutate(id = paste(symbol,year(date),mth = month(date)))
daily <- daily %>% mutate(traded = ifelse(is.na(turnover), FALSE, ifelse(turnover == 0,FALSE, TRUE)))       # check if traded (T / F vector)

# 4.2 Summarise days traded per month per company (if <10) then exlude these months returns
vol_sum <- daily %>% group_by(id) %>% summarise(days = sum(traded))
vol_sum <- vol_sum %>% mutate(vol_incl = days > 10)                 # logical vector

# 4.3 Calculate monthly amihud measure
daily <- daily %>% mutate(abs_ret = abs(Ri))                                # absolute daily returns
daily <- daily %>% mutate(measure_dy = abs_ret/turnover)                            # ratio abs daily returns / dollar volume trade
sigma_measure <- daily %>% group_by(id) %>% summarise(measure_m = sum(measure_dy, na.rm=T)) # sum measure_dy by coy/month
amihud_m <- sigma_measure %>% left_join(vol_sum, by='id') %>% filter(vol_incl==TRUE)        # add days traded + filter out <10 (or na)
amihud_m <- amihud_m %>% mutate(amihud = (1/days)*measure_m*1000000)                    # x10^6 as per Amihud (2002)

# 4.4 Chart amihud scores & checks
(amihud_chart <- amihud_m %>% ggplot(aes(x=amihud)) +
    geom_histogram(fill=iptblu) +
```

```
  geom_vline(xintercept = amihud_cutoff) +
  scale_y_continuous(expand = c(0,0)) +
  scale_x_continuous(limits = c(0,150)) +

# 4.5  Checks
sum(amihud_m$amihud>amihud_cutoff, na.rm=T)                # check for how many firm months fail measure
sum(amihud_m$amihud>amihud_cutoff, na.rm=T)/ nrow(monthly)   # prop of firm months which fail amihud
sum(is.na(amihud_m$amihud))                                # checks for NAs (should be 0)
# firm months to exclude based on amihud measure
fail_amihud <- amihud_m %>% filter(amihud > amihud_cutoff)   # amihud > 25 indicates illiquidity and should be removed

################################    5. Estimate model    ###################################################################
## Function to estimate market model: excess_i = alpha + B*excess_m + epsilon

# 5.1  Find the market associated with y
monthly <- monthly %>% left_join(select(info,symbol,name,country,indx), by='symbol')  # add market index code
# Get market returns for each month
monthly <- monthly %>% left_join(rm, by=c('date', 'indx'='mkt'))          # primary key becomes date + indx (mkt code)
monthly <- monthly %>% rename(Rm = return)                                # rename to avoid confusion
monthly <- monthly %>% left_join(vol_sum, by='id')                        # add vol_incl indicator
(monthly<- monthly %>% select(id, symbol, date, name, country, Rf_code, indx, Ri_val, turnover,days, vol_incl, Ri, Rf, Rx, Rm))

# 5.2  Apply Filters
monthly_adj <- monthly %>% filter(vol_incl == TRUE)                # Remove firm months where days_traded < 10
smp_10daysfilter <- nrow(monthly_adj %>% distinct(symbol))
monthly_adj <- monthly_adj %>% filter(!id %in% fail_amihud$id)     # Remove firm months with amihud above threshold (>25)
smp_amihudfilter <- nrow(monthly_adj %>% distinct(symbol))         # counts rows by symbol after liq adj
n2 <- data_frame(firms = n_distinct(monthly$symbol), filter = "< 10 trading days or exceed Amihud threshold")
```

```
# 5.2.1  Less than 3 years of eligible firm months
under3yrs   <- monthly_adj %>% group_by(symbol) %>% summarise(months = n()) %>% filter(months<36)
keep        <- !sample1 %in% under3yrs$symbol
sample_final<- sample1[keep]
under3yrs   <- under3yrs %>% mutate(Error_Type = "< 36 months") %>% select(Error_Type, symbol)
error_list  <- bind_rows(error_list, under3yrs)

# 5.2.2  Remove firms with < 36 months of eligible trading data
monthly_adj <- monthly_adj %>% filter(!symbol %in% under3yrs$symbol)
monthly_adj <- monthly_adj %>% filter(!is.na(Rm), !is.na(Rx))       # Remove NA from our sample
n3 <- data_frame(firms = n_distinct(monthly_adj$symbol), filter = 'At least 3 years of eligible trading data')

# 5.3  Run Model: For each company, regress excess returns on market returns to determine beta
params <- monthly_adj %>% group_by(symbol) %>% do(tidy(lm(Rx ~ Rm, data = .)))

## 5.4 Vasicek (beta) adjustment
# 5.4.1  Calculate sample variance of calculated betas
params <- params %>% mutate(std.var = std.error^2)
var_est_betas <- var(params$estimate)
# 5.4.2  Calculate lambda parameter
params <- params %>% mutate(lambda = var_est_betas/(var_est_betas + std.var))
# 5.4.3  Calculate vasicek: Bv=Bv_hat*(1-lambda)+lambda*B_ols (where Bv_hat = 1)
params <- params %>% mutate(vas = (1*(1-lambda))+(lambda*estimate))

## 5.5  Tidy up params
params <- params %>% mutate(Diff = vas - estimate)    #for plotting purposes
betas  <- params %>% filter(term == 'Rm') %>% arrange(desc(vas)) %>% select(-term)   # remove intercept terms
betas  <- betas %>% left_join(select(info, symbol, name, ds_beta), by="symbol") %>% tbl_df()
intercepts <- params %>% filter(!term == 'Rm')
```

```
names(betas) <- c('Symbol','OLS_Beta','Std.Error','Test_Statistic','P.Value','Std.Var','Lambda','Vas_Beta','Vas-OLS','Name','DS_Beta')

# 5.5.1  Vasicek averages
median_vas_beta <- round(median(betas$Vas_Beta),3)
mean_vas_beta   <- round(mean(betas$Vas_Beta),3)

################################    6. Unlever beta     ###############################################################
# 6.1  Add tax rates
betas <- betas %>% left_join(select(info, symbol, tax_rate), by=c('Symbol'='symbol'))

# 6.2 Get gearing ratios (debt to assets)
req     <- paste0(sample_final,"~=731~",eDate,"~:",eDate,"~D")
dat     <- ds(user, requests = req)
gearing <- lapply(dat["Data",], data.frame)
gearing <- list.stack(gearing, fill=TRUE) %>% tbl_df()
gearing <- gearing %>% rename(GEARING = X_x0037_31)

# 6.3  Convert to debt to equity
gearing <- gearing %>% mutate(DE = GEARING/(100-GEARING)) %>%
  mutate(Symbol = as.character(SYMBOL)) %>%
  select(Symbol, GEARING, DE)
# Add D/E to betas
betas <- betas %>% left_join(gearing, by='Symbol')

# 6.4  Unlever betas
# BU = BL / [1 + ((1 - Tax Rate) x Debt/Equity)]
betas <- betas %>% mutate(Unlev_Beta = Vas_Beta / (1+ ((1-tax_rate) * DE)))
# Re-lever betas at target gearing (60%)
betas <- betas %>% mutate(Relev_Beta = Unlev_Beta * (1 + ((1-tax_rate) * target_de)))
```

```
# Re-arrange
betas <- betas %>% filter(!is.na(GEARING)) %>%
  select(Symbol, Name, contains("Beta"), everything())

# Collect firm counts for various filter stages
n4 <- data_frame(firms = n_distinct(betas), filter = "No gearing data")
n_counts <- rbind(n0,n1,n2,n3,n4)

# 6.5  Median unlevered beta
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 gearing for the sample
median_gearing <- round(median(betas$GEARING, na.rm=T))
mean_gearing <- round(mean(betas$GEARING, na.rm=T))



################################       7. Output       ################################################################

# 7.1   Create output filename
tmp <- unlist(strsplit(file,split='\\\\')) #Breaks input template into character vector split on '\\'
tmp <- tmp[length(tmp)] #gets final element (the input template name), 'template_name.xlsx'
tmp <- unlist(strsplit(tmp,split='\\.')) #breaks template into name and '.xlsx'
savename <- paste0(tmp[1],'_',format(Sys.time(),'%Y-%m-%d_%I-%M-%p')) #creates filestamp
#NB: Must use format in Sys.time(), otherwise cannot save files with : characters
rm(tmp)

# 7.2  Spreadsheet with various output data (models, insufficient volume, high Amihud)
sheets <- list('Model Output'=betas,'Intercepts'=intercepts, 'Errors'=error_list)
```

```
write.xlsx(sheets,file=paste0('Output_',savename,'.xlsx'))

# 7.3  R outputs for Markdown doc
save(daily, monthly, amihud_m, betas, error_list, n_counts, file = "Beta Estimation Data.RData")

# Long version
betas_long <- betas %>% select(Symbol:Relev_Beta) %>%
  gather(Beta_Type, Value, OLS_Beta:Relev_Beta)

#  7.4  Dotplot1: Unlevered -> Relevered
(dotplot2 <- 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') +

    theme_ipart +
    labs(x= 'Relevered beta after vasicek adjustment',
         y= 'Company code'))
ggsave(dotplot1, file=paste0('relev_dotplot_',savename,'.png'), width = 10, height = 10)



# 7.5  Dotplot2 - OLS -> Vasicek adjustment
(dotplot <- 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') +
    annotate('text',x= median_vas_beta,y=1.5,label=paste0('Median vasicek beta = ',median_vas_beta), col='red',size=6) +
    theme_ipart +
    labs(x= 'Beta after Vaiscek adjustment', #Vasicek
         y= 'Company code'))

(dotplot <- 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_vas_beta, col='red') +
    annotate('text',x= median_vas_beta +0.5,y=1.5,label=paste0('Median vasicek beta = ',median_vas_beta), col='red',size=6) +
    theme_ipart +
    labs(x= 'Beta after Vasicek adjustment', #Vasicek
         y= 'Company code'))

ggsave(dotplot2, file=paste0('dotplot_',savename,'.png'), width=10, height=10)

ggsave(dotplot, file=paste0('dotplot_',savename,'.png'), width=10, height=10)
```