Análise de Carteiras usando o R - Parte 3

Bibliografia – BKM, cap. 7

Claudio Lucinda

FEA-RP/USP

Portifólios Arriscados ótimos

Importando dados do IBOVESPA

- Para começarmos a analisar, precisamos coletar os dados.
- Vou mostrar para vocês como baixar os dados de fechamento diretamente da Internet – Yahoo Finance.
- Vou mostrar dois jeitos de fazer isso. A primeira é usando o pacote `quantmod` e a segunda é usando um pacote desenvolvido por um brasileiro, o Marcelo Perlin, o `BatchGetSymbols`.
- Só um cuidado: os dados do Yahoo Finance tem problemas com relação à desdobramentos (*stock splits*) e pagamentos de dividendos/Juros sobre Capital Próprio.
    - Ou seja, cuidado com o uso destes dados no seu trabalho final. Melhor importar do Economatica, que oferece esses dados completamente ajustados para esses eventos

Importando dados do IBOVESPA

- O código com os comandos está em `Ibov_data.R`.
- Instalando os pacotes:

```
library(BatchGetSymbols)
library(PortfolioAnalytics)
library(xlsx)
library(tidyverse)
library(quantmod)
library(tidyquant)
```

Composição do Ibovespa

- A composição mais recente da carteira pode ser encontrada no site a seguir:
- http://www.bmfbovespa.com.br/pt_br/produtos/indices/indices-amplos/indice-ibovespa-ibovespa-composicao-da-carteira.htm

```
rm(list=ls())
graphics.off()
IBOV_Data<-read.xlsx("./Comp_IBOV.xlsx",sheetName="IBOV",
                     stringsAsFactors=F)
tickers<-data.frame(IBOV_Data$Ticker,
                    stringsAsFactors=FALSE)
tickers_Yah<-paste0(tickers[[1]],".SA")[1:64]
datainicial<-"2004-01-01"
datafinal<-"2018-02-23"
```

As Units

- Existem cinco ativos que não conseguimos baixar do Yahoo Finance.
  - KLBN11.SA, SANB11.SA, SAPR11.SA, TAEE11.SA e VVAR11.SA
- São as chamadas *Units*
  - Units são ativos compostos por mais de uma classe de valores mobiliários, como uma ação ordinária e um bônus de subscrição, por exemplo, negociados em conjunto.
  - As units são compradas e/ou vendidas no mercado como uma unidade.

Definição das Units

Empresa	Ticker	Definição
Klabin	KLBN11	1 ação ON + 4 ações PN
SANTANDER BR	SANB11	1 ação ON + 1 ação PN
SANEPAR	SAPR11	1 ação ON + 4 ações PN
TAESA	TAEE11	1 ação ON + 2 ações PN
VIAVAREJO	VVAR11	1 ação ON + 2 ações PN

Baixando os dados pelo quantmod

```
dataEnv <- new.env()
getSymbols(tickers_Yah[1:35],auto.assign=TRUE,
           from=datainicial,to=datafinal,env=dataEnv)
getSymbols(tickers_Yah[37:52],auto.assign=TRUE,
           from=datainicial,to=datafinal,env=dataEnv)
getSymbols(tickers_Yah[54:56],auto.assign=TRUE,
           from=datainicial,to=datafinal,env=dataEnv)
getSymbols(tickers_Yah[58:62],auto.assign=TRUE,
           from=datainicial,to=datafinal,env=dataEnv)
getSymbols(tickers_Yah[64],auto.assign=TRUE,
           from=datainicial,to=datafinal,env=dataEnv)
plist <- eapply(dataEnv, Ad)
IBOV_Cots <- do.call(merge, plist)
```

Baixando os dados pelo BatchGetSymbols

```
df_IBOV<-BatchGetSymbols(tickers_Yah,
                         first.date = datainicial,
                         last.date = datafinal,
                         thresh.bad.data = .75)
data_IBOV<-reshape.wide(df_IBOV$df.tickers)$price.close
```

Limpando os dados

- Como disse antes, os dados do Yahoo Finance possuem problemas com o ajustamento das cotações para eventos como dividendos, *stock splits* e Juros Sobre Capital Próprio.
- Esse é um trabalho que dá pra ser automatizado, mas aqui vou fazer um ajuste meio simples apenas para fins didáticos.
- Várias funções, para transformar em `xts` os objetos, para limpar essas observações e tirar os `na`.

Funções de Limpeza 01

```
conv_ts <- function(.df) {
  OUT<-xts(.df[,-1],order.by=.df[,1])
  return(OUT)
}

clean_na <- function(DF) {
  OUT<-DF[, sapply(DF, function(x)
    sum(is.na(x)))!=nrow(DF)]

  return(OUT)
}
```

Funções de Limpeza 02

```
remove_outliers <- function(x, .pct=.25,
                                iqr.tresh=1.5,
                                na.rm=TRUE, ...) {
  .pctsup<-1-.pct
  qnt <- quantile(x, probs=c(.pct, .pctsup),
                  na.rm = na.rm, ...)
  H <- iqr.tresh * IQR(x, na.rm = na.rm)
  y <- x
  y[x < (qnt[1] - H)] <- NA
  y[x > (qnt[2] + H)] <- NA
  y
}
```

Funções de Limpeza 03

```
df_returns <- function(.df) {
  df_temp<-.df
  for (nom in colnames(.df)) {
    df_temp[,nom]<-Return.calculate(df_temp[,nom])
  }
  return(df_temp)
}
```

Funções de Limpeza 04

```
df_monthly <- function(.df) {
  df_0<-to.monthly(.df[,1])[,4]
  colnames(df_0)[1]<-colnames(.df)[1]
  len<-length(colnames(.df))
  for (nom in seq(2,len)) {
    df_0<-merge(df_0,to.monthly(.df[,nom])[,4])
    colnames(df_0)[nom]<-colnames(.df)[nom]
  }
  return(df_0)
}
```

Limpando os dados

```
data_IBOV<-conv_ts(data_IBOV)
data_IBOV<-remove_outliers(data_IBOV,pct=.25,iqr.tresh = 3)
data_IBOV<-clean_na(data_IBOV)

IBOV_Returns<-df_returns(data_IBOV)
data_IBOV_monthly<-df_monthly(data_IBOV)

IBOV_Returns_monthly<-df_returns(data_IBOV_monthly)
IBOV_Returns_Final<-na.omit(IBOV_Returns_monthly)

save(list=ls(),file="IbovData.RDS")
```

Carteiras com dois ativos

Carteiras com dois ativos

- Vamos mostrar como calcular o retorno e o desvio-padrão para uma carteira com dois ativos.
- Notações:
  - $\bar{r_A}$, σ_A, w_A - Retorno Esperado, Desvio-Padrão dos retornos e peso do ativo A na carteira.
  - $\bar{r_B}$, σ_B, w_B - Retorno Esperado, Desvio-Padrão dos retornos e peso do ativo A na carteira. Por enquanto, vamos supor que $w_B = 1 - w_A$.
  - ρ_{AB}, σ_{AB} - Coeficiente de correlação e covariância entre os retornos dos ativos A e B.

$$\rho_{AB} = \frac{\sigma_{AB}}{\sigma_A \sigma_B}$$

Carteiras com dois ativos - Média e Desvio-Padrão

- Retorno esperado:

$$\bar{r_P} = w_A \bar{r_A} + (1 - w_A)\bar{r_B}$$

- Desvio-Padrão

$$\sigma_P^2 = w_A^2 \sigma_A^2 + (1 - w_A)^2 \sigma_B^2 2 w_A (1 - w_A) \sigma_{AB}$$

- Desvio-Padrão - em função do coeficiente de correlação:

$$\sigma_P^2 = w_A^2 \sigma_A^2 + (1 - w_A)^2 \sigma_B^2 + 2 w_A (1 - w_A) \sigma_A \sigma_B \rho_{AB}$$

- O Termo relevante aqui é o ρ_{AB}.

Fazendo isso no R

- A implementação disso está no arquivo `TwoAsset.R`
- Esse não precisa de nenhuma library instalada. Só com os códigos base.
- Inicialmente implementamos isso como uma função:

```
two_asset <- function(.w,.rho,.mu.A,.mu.B,.sig.A,.sig.B) {
  sig2.A = .sig.A^2
  sig2.B = .sig.B^2
  sig.AB <- .rho*.sig.A*.sig.B
  mu.p1 <- .w*.mu.A + (1-.w)*.mu.B
  sig2.p1 <- (.w^2) * sig2.A + ((1-.w)^2)* sig2.B +
    2*.w*(1-.w)*sig.AB
  sig.p1 <- sqrt(sig2.p1)
  w0<-1
  VaR.p1 <- (mu.p1 + sig.p1*qnorm(0.05))*w0
  results<-data.frame(mu.p1,sig.p1,VaR.p1)
}
```

Carteiras com diferentes valores de ρ_{AB}

```
port_1<-plyr::ldply(seq(0,1,.01),
                    function(x)
                      two_asset(x,.rho=1,
                                .mu.A=.175,.mu.B=.055,
                                .sig.A=.258,.sig.B=.115))
port_0<-plyr::ldply(seq(0,1,.01),
                    function(x)
                      two_asset(x,.rho=0,
                                .mu.A=.175,.mu.B=.055,
                                .sig.A=.258,.sig.B=.115))

port_minus_1<-plyr::ldply(seq(0,1,.01),
                          function(x)
                            two_asset(x,.rho=-1,
                            .mu.A=.175,.mu.B=.055,
                            .sig.A=.258,.sig.B=.115))
```

Ajeitando tudo

```
new_data<-merge(port_1,port_0,by="mu.p1")
colnames(new_data)<-c("mu.p1","sig_1",
                      "Var_1","sig_2","Var_2")
new_data<-merge(new_data,port_minus_1,by="mu.p1")
colnames(new_data)<-c("mu.p1","sig_1",
                      "Var_1","sig_2","Var_2","sig_3",
                      "Var_3")


library(ggplot2)
```

Gráfico - Comando

```
ggplot() +geom_path(data=new_data,aes(x=sig_1,y=mu.p1),
                    color="red") +
  geom_path(data=new_data,aes(x=sig_2,y=mu.p1),
            color="blue") +
  geom_path(data=new_data,aes(x=sig_3,y=mu.p1),
            color="green") +
  labs(x = "Std. Dev.", y = "Exp. Return") +
  scale_x_continuous(limits=c(0,.3)) +
  geom_point(aes(x=.258,y=.175), col="blue") +
  geom_point(aes(x=.115,y=.055), col="red")
```

Gráfico