**Apêndice Aula 4:** [Hipóteses do Modelo de Regressão Multivariado & O Modelo em Notação Matricial – 12/09 (n) / 14/09 (v)](https://edisciplinas.usp.br/course/view.php?id=103128#section-5)

**Códigos Úteis no R**

**- Instalação de pacotes e abertura**

[install.packages](https://rdrr.io/r/utils/install.packages.html)("tidylog")

library("tidyverse")

library("tidylog")

**- Abrindo Dados em formato .csv**

Provavelmente a função mais usada é read\_csv, que abre arquivos com extensão .csv. Em vez de abrir arquivos do nosso computador, também podemos abrir links online e, para simplificar, disponibilizamos vários arquivos em nosso repositório do github. Para começar, vamos salvar o url do link de um arquivo .csv como um objeto, file1, e abri-lo.

file1 < "https://raw.githubusercontent.com/leobarone/ifch\_intro\_r/master/data/bf\_amostra\_hv.csv"

dados <- read\_csv(file1)

A função read\_delim nos dá mais flexibilidade para lidar com tipos incomuns de arquivos. Por exemplo, podemos especificar o caractere usado para separar as colunas (por padrão, esta é a vírgula em read\_csv, mas muitos dados brasileiros usam “;” como padrão, já que a vírgula é, no Brasil, um separador decimal):

dados <- read\_delim(file1,

delim = ",")

# ou

file\_semi\_colon <- "https://raw.githubusercontent.com/leobarone/ifch\_intro\_r/master/data/bf\_amostra\_hp.csv"

dados <- read\_delim(file\_semi\_colon,

delim = ";")

**- Selecionar Variáveis (Select)**

Para selecionar um subconjunto das variáveis (colunas) para manter no tibble, usamos o verbo select(). Começamos com o tibble, o pipe, e finalmente o verbo, select:

banco\_de\_dados %>% select(var1, var2, var3)

**- Transformar Variáveis (Mutate)**

Especificar o cálculo a ser realizado usando os nomes de colunas atuais, e o nome da nova variável (coluna) em que vamos guardar o resultado do cálculo. Por exemplo, se queremos dobrar a var2:

banco\_de\_dados <- banco\_de\_dados %>% mutate(var2\_dobro= var2\*2)

**- Matrizes**

[**https://rpubs.com/adelmofilho/AlgebraMatricial#:~:text=Em%20R%2C%20a%20matriz%20%C3%A9,n%C3%BAmero%20de%20linhas%20e%20colunas**](https://rpubs.com/adelmofilho/AlgebraMatricial#:~:text=Em%20R%2C%20a%20matriz%20%C3%A9,n%C3%BAmero%20de%20linhas%20e%20colunas)**.**

Em R, a matriz é uma estrutura de dados homogênea (deve conter mesmo tipo de dados, e.i. numérico, caracteres, lógico) de duas dimensões. Sua sintaxe básica é realizada por meio da função matrix(), tendo como argumentos os elementos que compõem a matriz, na forma de um vetor, e o número de linhas e colunas.

m <- matrix(data = 1:9, nrow = 3, ncol = 3)

print(m)

Por *default* os elementos são organizados por coluna na matriz, caso se deseje que estes sejam organizados por linha usa-se o argumento byrow = TRUE.

m2 <- matrix(data = 1:9, nrow = 3, ncol = 3, byrow = TRUE)

print(m2)

Para criar uma matriz contendo um único valor, e.g. matriz nula, informa-se este valor no argumento data em conjunto com o número de linhas e colunas.

m4 <- matrix(data = 0, nrow = 3, ncol = 5)

print(m4)

Outra possibilidade é criar vetores e, em seguida, juntá-los em uma matriz:

x <- c(1, 2, 3)

x2 <- c(4, 5, 6)

x3 <- c(7, 8, 9)

matriz <- cbind(x, x2, x3)

print(matriz)

**Operações matriciais: Soma**

m\_soma <- M7 + M6

print(m\_soma)

**Operações matriciais: subtração**

m\_subt <- M7 - M6

print(m\_subt)

**Operações matriciais: Multiplicação por escalar**

Uma matriz de dimensionalidade qualquer quando multiplicada por um escalar **k**, tal que k ∈ R, resulta em uma matriz de mesmas dimensões cujos elementos são o produto do escalar k por cada um dos elementos da matriz original.

m\_prod\_escalar <- 42 \* M9

print(m\_prod\_escalar)

**Operações matriciais: Multiplicação matricial**

Em R, a multiplicação matricial clássica é desenvolvida pelo uso do operador %\*%, respeitando a equidade entre o número de colunas da matriz que pré-multiplica e o número de linhas da matriz que pós-multiplica.

m\_prod = M6 %\*% M8

print(m\_prod)

**Transposta de uma matriz**

Em R, a transposta de uma matriz é determinada pela pela função t().

m\_transp = t(M8)

print(m\_transp)

**Determinante de uma matriz**

O determinante de uma matriz é uma função matricial que converte uma matriz quadrada em um escalar. Em R, o determinante de uma matriz é calculado pela função det()

det(m8)

**Inversa de uma matriz**

A inversa de uma matriz é estimada pela função solve(), mesma função utilizada para resolução de sistemas de equações em R.

m\_inversa = solve(M6)

print(m\_inversa)

Facilmente, verifica-se a condição de existência da matriz inversa:

solve(m6)%\*%m6

**- Regressão**

Regressão mais simples, um modelo linear, apropriado para variáveis dependentes contínuas. A função é lm(), e indicando data=. podemos passar os nossos dados diretamente para ela no fluxo de pipe. É só inserir a fórmula que define a nossa regressão, como sempre com a variável dependente na esquerda e as variáveis independentes (explicativas) na direita. Por exemplo:

<tabela> %>% [lm](https://rdrr.io/r/stats/lm.html)(<var\_dependente> ~ <var\_independente>, data=.)

Para regressão composta, adicionar o sinal + e a outra variável explicativa.

<tabela> %>% [lm](https://rdrr.io/r/stats/lm.html)(<var\_dependente> ~ <var\_independente>

+ <var\_independente\_2>, data=.)

Comparar dois modelos de regressão na mesma tabela, é só salvar as duas regressões como objetos, e encaminhar eles juntos na forma de uma list() para stargazer:

[install.packages](https://rdrr.io/r/utils/install.packages.html)('stargazeR')

[library](https://rdrr.io/r/base/library.html)(stargazer)

reg1 <- <tabela> %>% [lm](https://rdrr.io/r/stats/lm.html)(var\_dep ~ var\_ind, data=.)

reg2 <- <tabela> %>% [lm](https://rdrr.io/r/stats/lm.html)(var\_dep ~ var\_ind + var\_ind\_2, data=.)

[list](https://rdrr.io/r/base/list.html)(reg1, reg2) %>% [stargazer](https://rdrr.io/pkg/stargazer/man/stargazer.html)(type="html")

**- Resíduos e R2**

Use a função summary() para gerar uma tabela de regressão (com coeficientes e desvios padrões dos coeficientes) e várias estatísticas, incluindo resíduos.

<tabela> %>% [lm](https://rdrr.io/r/stats/lm.html)(var\_dep ~ var\_ind, data=.) %>% [summary](https://rdrr.io/r/base/summary.html)()

Para extrair especificamente os resíduos e R², podemos usar:

regressao <- <tabela> %>% [lm](https://rdrr.io/r/stats/lm.html)(var\_dep ~ var\_ind, data=.)

# Resíduos:

regressao$residuals

# R²

summary(regressao)$r.squared

Previsões e resíduos de regressões entre o valor atual e o valor previsto: função augment() (do pacote broom) - começando com o modelo rodado acima, augment() gera um novo tibble com dados originais usados na regressão, mais as colunas adicionais com os resultados.

[install.packages](https://rdrr.io/r/utils/install.packages.html)('[augment](https://generics.r-lib.org/reference/augment.html)')

library([augment](https://generics.r-lib.org/reference/augment.html))

reg1 %>% [augment](https://generics.r-lib.org/reference/augment.html)()

**- Matriz de Covariância**

Para gerar a matriz de covariâncias no R, utilizamos a função cov()

cov(objeto)

**- Correlação**

A geração da matriz de correlação em R se dá pela função cor():

cor(objeto)

**- Gráfico de dispersão**

A importação do pacote tidyverse já torna o pacote ggplot disponível para uso. Para visualizar a dispersão de duas variáveis em um plano bidimensional, usamos:

ggplot(<nome\_tabela>, aes(<variavel\_eixo\_x>, <variavel\_eixo\_y>) +

geom\_point()