Métodos Quantitativos

Prof. Dr. A. L. Korzenowski

Introdução ao R

O que é o R?

Um ambiente que integra diversos programas para computação estatística e composição de gráficos.

- Permite manipulação e armazenamento de dados.
- Oferece um conjunto de operadores para cálculos sobre vetores e matrizes.
- Coloca à disposição do usuário uma grande coleção de ferramentas para análise de dados.
- Comunicação por meio de uma linguagem simples e eficaz, similar à linguagem S.

O R pode ser obtido a partir de algum *mirror* no seu site oficial *The R Project for Statistical Computing*

Se foi feita a instalação completa do R, é muito provável que esteja disponível uma vasta documentação própria do R.

- Perguntas freqüentes do R
- Perguntas freqüentes do R para ambiente Windows
- Manuais em Portable Document Format (PDF)
- Ajuda online via comando help( ).
- Ajuda em HTML (com Introdução, Busca, Pacotes, Linguagem R, Instalação e Administração, etc.)

Um comando importante – fundamental – é o help( ):

- help(FUNCAO)
- help("ASSUNTO")

Recentemente, a comunidade desenvolveu o programa RStudio, que adiciona uma série de funcionalidades visuais ao ambiente de programação, na busca de torná-lo mais amigável. O RStudio é um ambiente de desenvolvimento integrado (IDE) para R. Ele inclui um console, editor de realce de sintaxe que suporta execução direta de código, além de ferramentas para plotagem, histórico, depuração e gerenciamento de espaço de trabalho.

Utilizaremos a opção RStudio Cloud, uma plataforma on-line onde é necessário apenas um navegador e acesso a internet. O RStudio, seja na versão local ou na Cloud, tem a aparência apresentada na Figura 1. Esta imagem foi recortada do RStudio IDE CHEAT SHEET, que você pode baixar no link https://github.com/rstudio/cheatsheets/raw/master/rstudio-ide.pdf.

Tarefa

1. Acessar o RStudio Cloud no link http://rstudio.cloud.
2. Fazer o cadastro.
3. Criar um projeto para a disciplina no ambiente.
4. Explorar os espaços e utilizar os primeiros comandos.

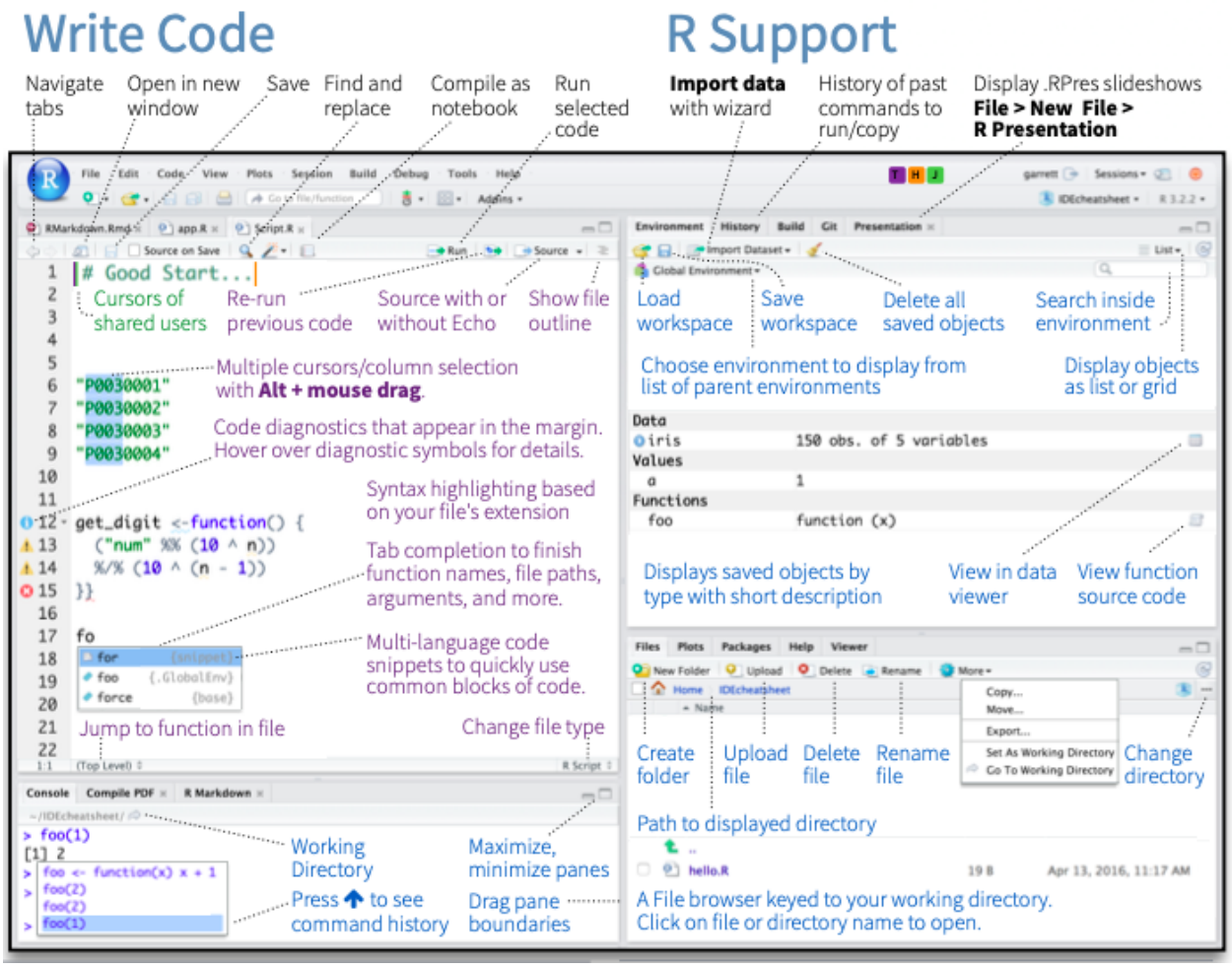


Figure 1: RStudio IDE CHEAT SHEET

Introdução ao *software* e linguagem R

O ideal para aprender a usar o R é "usá-lo!". Então, a melhor forma de se familiarizar com os comandos do R é ler um texto introdutório e ao mesmo tempo ir digitando os comandos no R e observando os resultados, gráficos, etc. Aprender a usar o R pode ser difícil e trabalhoso, mas lembre-se, o investimento será para você!

Para usar o R é necessário conhecer e digitar comandos. Alguns usuários acostumados com outros programas notarão de início a falta de "menus" (opções para clicar). Na medida em que utilizam o programa, os usuários (ou boa parte deles) tendem a preferir o mecanismo de comandos, pois é mais flexível e com mais recursos. Algumas pessoas desenvolveram módulos de "clique-clique" para o R, como o R-commander. Porem, eu acredito que ao usar um módulo de "clique-clique" perdemos a chance de aprender uma das maiores potencialidades e virtudes do R, que é a programação.

O R é case-sensitive, isto é, ele diferencia letras maiúsculas de minúsculas, portanto A é diferente de a. O separador de casas decimais é ponto ".". A vírgula é usada para separar argumentos (informações). Não é recomendado o uso de acentos em palavras (qualquer nome que for salvar em um computador, não só no R, evite usar acentos. Acentos são comandos usados em programação e podem causar erros, por exemplo, em documentos do word e excel).

O R é um programa leve (ocupa pouco espaço e memória) e geralmente roda rápido, até em computadores não muito bons. Isso porque ao instalarmos o R apenas as configurações mínimas para seu funcionamento básico são instaladas (pacotes que vem na instalação "base"). Para realizar tarefas mais complicadas pode ser necessário instalar pacotes adicionais (packages). Não basta apenas instalar um pacote. Para usá-lo é necessário "carregar" o pacote sempre que você abrir o R e for usá-lo. Use a função library para rodar um pacote.

No R existe um comando que mostra como citar o R ou um de seus pacotes. Veja como fazer:

```
citation()
```

```
##
## To cite R in publications use:
##
##   R Core Team (2019). R: A language and environment for statistical
##   computing. R Foundation for Statistical Computing, Vienna, Austria.
##   URL https://www.R-project.org/.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {R: A Language and Environment for Statistical Computing},
##     author = {{R Core Team}},
##     organization = {R Foundation for Statistical Computing},
##     address = {Vienna, Austria},
##     year = {2019},
##     url = {https://www.R-project.org/},
##   }
##
## We have invested a lot of time and effort in creating R, please cite it
## when using it for data analysis. See also 'citation("pkgname")' for
## citing R packages.
```

Para citar um pacote, por exemplo psych, basta colocar o nome do pacote entre aspas:

```r
citation("psych")
```

```
##
## To cite the psych package in publications use:
##
##   Revelle, W. (2018) psych: Procedures for Personality and
##   Psychological Research, Northwestern University, Evanston, Illinois,
##   USA, https://CRAN.R-project.org/package=psych Version = 1.8.12.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {psych: Procedures for Psychological, Psychometric, and Personality Research},
##     author = {William Revelle},
##     organization = { Northwestern University},
##     address = { Evanston, Illinois},
##     year = {2018},
##     note = {R package version 1.8.12},
##     url = {https://CRAN.R-project.org/package=psych},
##   }
```

R como calculadora

O forma de uso mais básica do R é usá-lo como calculadora. Os operadores matemáticos básicos são: + para soma, - subtração, * multiplicação, / divisão e ^ exponenciação. Digite as seguintes operações na linha de comandos do R:

```r
2+3
2*3
2/3
2^3
```

O R tem diversas funções que podemos usar para fazer os cálculos desejados. O uso básico de uma função é escrever o nome da função e colocar os argumentos entre parênteses, por exemplo: **função(argumentos)**. **função** especifica qual função irá usar e **argumentos** especifica os argumentos que serão avaliados pela função. Não se assuste com esses nomes, com um pouco de pratica eles se tornarão triviais. Rode as seguintes linhas de código no R para comprender melhor o uso de funções:

```r
sqrt(9)       # Extrai a raiz quadrada dos argumentos entre parênteses
sqrt(2*3^2)   # Extrai a raiz quadrada de 18
sqrt((2*3)^2) #Extrai a raiz quadrada de 36
seq(from = 1, to = 5, by = 1 )  # Gera uma sequência de um até 5 em intervalo regular
prod(1,2,3,4) # Determina o produtório 1x2x3x4
```

Objetos do R (O que são?):

O que são os Objetos do R? Existem muitos tipos de objetos no R que só passamos a conhecê-los bem com o passar do tempo. Por enquanto vamos aprender os tipos básicos de objetos.

1. vetores: uma seqüência de valores numéricos ou de caracteres (letras, palavras).

2. matrizes: coleção de vetores em linhas e colunas, todos os vetores dever ser do mesmo tipo (numérico ou de caracteres).
3. dataframe: O mesmo que uma matriz, mas aceita vetores de tipos diferentes (numérico e caracteres). Geralmente nós guardamos nossos dados em objetos do tipo data frame, pois sempre temos variáveis numéricas e variáveis categóricas (por exemplo, largura do rio e nome do rio, respectivamente).
4. listas: conjunto de vetores, dataframes ou de matrizes. Não precisam ter o mesmo comprimento, é a forma que a maioria das funções retorna os resultados.
5. funções: as funções criadas para fazer diversos cálculos também são objetos do R. No decorrer da apostila você verá exemplos de cada um destes objetos.

Algumas funções do R possuem demonstrações de uso. Vejamos alguns exemplos:

```
demo(graphics)
demo(persp)
```

Como criar objetos?

O comando <- (sinal de menor e sinal de menos) significa assinalar (assign). Indica que tudo que vem após este comando será salvo com o nome que vem antes. Se quisermos atribuir um vetor de dados ao nome **x** para cálculos posteriores, podemos fazê-lo da seguinte forma:

```
x <- c(2,3,4,5,6,9,12,14,16,17,21,24)
```

onde **x** é o nome atribuído ao vetor e **c** a função concatenar que agrupa os dados entre parênteses dentro do objeto que será criado. Para ver os valores (o conteúdo de um objeto), basta digitar o nome do objeto na linha de comandos.

```
##  [1]  2  3  4  5  6  9 12 14 16 17 21 24
```

Para fazer operações com objetos vetorias, existem funções úteis, como por exemplo algumas funções estatísticas.

```
length(x) # Tamanho do vetor
```

```
## [1] 12
```

```
min(x)    # Mínimo valor de x
```

```
## [1] 2
```

```
max(x)    # Máximo valor de x
```

```
## [1] 24
```

```
mean(x)   # Média aritmética de x
```

```
## [1] 11.08333
```

```r
sd(x)      # Desvio-padrão de x
```

```
## [1] 7.378819
```

Se quiser usar estas informações posteriormente, basta salvá-las em um objeto no ambiente que ele poderá ser chamado a qualquer tempo.

```r
sum(x)
```

```
## [1] 133
```

```r
media <- mean(x)
n <- length(x)
media
```

```
## [1] 11.08333
```

```r
n
```

```
## [1] 12
```

```r
media * n
```

```
## [1] 133
```

Acessar valores dentro de um objeto [colchetes]

Caso queira acessar apenas um valor do conjunto de dados use colchetes [ ]. Isto é possível porque o R salva os objetos como vetores, ou seja, a sequencia na qual você incluiu os dados é preservada. Por exemplo, vamos acessar o quinto valor do objeto **x**.

```r
x[5]
```

```
## [1] 6
```

ou os valores de ordem 3, 5 e 7...

```r
x[c(3,5,7)]
```

```
## [1]  4  6 12
```

Se deseja substituir ou excluir um valor, proceda da seguinta forma:

```r
x
```

```
##  [1]  2  3  4  5  6  9 12 14 16 17 21 24
```

```r
x[10] <- 99   # Altera o décimo valor para 99
x
```

```
##  [1]  2  3  4  5  6  9 12 14 16 99 21 24
```

```r
x[-10]        # Note que o décimo valor (99) não aparece
```

```
##  [1]  2  3  4  5  6  9 12 14 16 21 24
```

Transformar dados

Em alguns casos é necessário, ou recomendado, que você transforme seus dados antes de fazer suas análises. Transformações comumente utilizadas são log e raiz quadrada.

```r
sqrt(x)   # Raiz quadrada dos valores de x
```

```
##  [1] 1.414214 1.732051 2.000000 2.236068 2.449490 3.000000 3.464102 3.741657
##  [9] 4.000000 9.949874 4.582576 4.898979
```

```r
log10(x)  # log(x) na base 10, apenas
```

```
##  [1] 0.3010300 0.4771213 0.6020600 0.6989700 0.7781513 0.9542425 1.0791812
##  [8] 1.1461280 1.2041200 1.9956352 1.3222193 1.3802112
```

```r
log(x)    # logaritmo natural de x
```

```
##  [1] 0.6931472 1.0986123 1.3862944 1.6094379 1.7917595 2.1972246 2.4849066
##  [8] 2.6390573 2.7725887 4.5951199 3.0445224 3.1780538
```

Para salvar os dados transformados dê um nome ao resultado. Por exemplo:

```r
x.log<-log10(x) # salva um objeto com os valores de aves em log
```

Gerar dados aleatórios

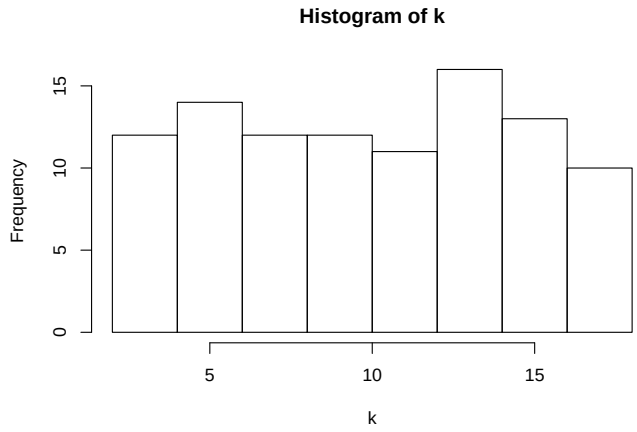
1. Gerar dados aleatórios com distribuição uniforme

- runif(n, min=0, max=1) gera uma distribuição uniforme com n valores, começando em min e terminando em max.

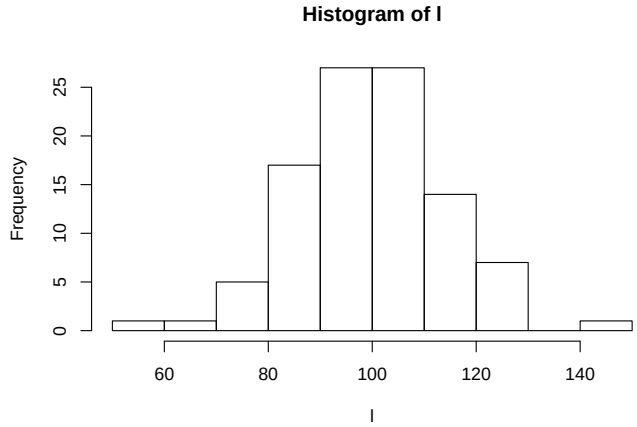
2. Gerar dados aleatórios com distribuição normal

- rnorm(n, mean=0, sd=1) gera n valores com distribuição normal, com média 0 e desvio padrão 1.

```r
set.seed(42)      # Fixa semente aleatória para reprodutibilidade
k <- runif(n = 100, min = 2, max = 17)
l <- rnorm(n = 100, mean = 100, sd = 15)
hist(k)           # Histograma dos dados k
```




```
hist(l)           # Histograma dos dados l
```



Veja o help da função **?Distributions** para conhecer outras formar de gerar dados aleatórios com diferentes distribuições.

Selecionar amostras aleatórias, ordenar e atribuir postos (ranks) aos dados

A função sample é utilizada para realizar amostras aleatórias:

sample(x, size=1, replace = FALSE) onde x é o conjunto de dados do qual as amostras serão retiradas, size é o número de amostras e replace é onde você indica se a amostra deve ser feita com reposição (TRUE) ou sem reposição (FALSE). Assim, **sample(1:10,5)** seleciona 5 dados com valores entre 1 e 10. Como não especificamos o argumento replace o padrão é considerar que a amostra é sem reposição (= FALSE). Com a função sample nós podemos criar varios processos de amostragem aleatória. Por exemplo, vamos criar uma moeda e "jogá-la" para ver quantas caras e quantas coroas saem em 10 jogadas.

```
moeda<-c("CARA","COROA") # primeiro criamos a moeda
sample(moeda,10,replace=TRUE)
```

```
##  [1] "CARA"  "COROA" "COROA" "COROA" "CARA"  "CARA"  "COROA" "CARA"  "CARA"
## [10] "COROA"
```

A função sort coloca os valores de um objeto em ordem crescente ou em ordem decrescente.

```r
set.seed(42)
exemplo<-sample(1:100,10)
exemplo
```

```
##  [1]  49  65  25  74  18 100  47  24  71  89
```

```r
sort(exemplo) # para colocar em ordem crescente
```

```
##  [1]  18  24  25  47  49  65  71  74  89 100
```

```r
sort(exemplo, decreasing=TRUE) # para colocar em ordem decrescente
```

```
##  [1] 100  89  74  71  65  49  47  25  24  18
```

A função order retorna a posição original de cada valor do objeto **exemplo** caso os valores do objeto **exemplo** sejam colocados em ordem.

```r
order(as.array(exemplo) )
```

```
##  [1]  5  8  3  7  1  2  9  4 10  6
```

Note que o primeiro valor acima é 5, isso indica que se quisermos colocar o objeto **exemplo** em ordem crescente o primeiro valor deverá ser o quinto valor do **exemplo**, que é o valor 18 (o menor deles).

```r
order(exemplo,decreasing=TRUE)
```

```
##  [1]  6 10  4  9  2  1  7  3  8  5
```

É importante entender o comando order, pois ele é muito usado para colocar uma planilha de dados seguindo a ordem de alguma de suas variáveis.

A função rank atribui postos aos valores de um objeto.

```r
exemplo # apenas para relembrar os valores do exemplo
```

```
##  [1]  49  65  25  74  18 100  47  24  71  89
```

```r
rank(exemplo) # Para atribuir postos (ranks) aos valores do exemplo
```

```
##  [1]  5  6  3  8  1 10  4  2  7  9
```

Veja que 100 é o maior valor do exemplo, portanto recebe o maior rank, no caso 10.

Importar conjunto de dados para o R

A tarefa de importar conjuntos de dados na versão nativa do R sempre foi um tanto desafiadora. Funções e argumentos além da necessidade de instalar e carregar pacotes especificos se os dados não estavam no formato adequado. Com o RStudio, a tarefa ficou muito simplificada pela facilidade de utilizar os menus - algo tipo *Click-and-Play*!

Crie um arquivo em Excel e explore o ambiente do RStudio. Na aba Environment, utilize a opção Import Dataset. Os pacotes necessários para carregar a sua planilha serão instalados e carregados pelo RStudio. LEMBRE-SE DE COMO DEVE SER CONSTRUÍDA UMA BASE DE DADOS PARA ANÁLISE!!!

Para selecionar (extrair) apenas partes do nosso conjunto de dados usando [ ] colchetes.

O uso de colchetes funciona assim: [linhas, colunas], onde está escrito linhas você especifica as linhas desejadas, na maioria dos casos cada linha indica uma unidade amostral. Onde está escrito colunas, você pode especificar as colunas (atributos) que deseja selecionar.

Se você ainda não carregou uma matriz de dados para o R, vamos criar uma **data.frame** de dados aleatórios com a função **matrix** e selecionar parte dos dados como exemplo. Na sua planilha carregada funciona da mesma forma.

```
dados <- as.data.frame(matrix(data = rnorm(n=20, mean = 20, sd = 5), ncol = 5))
dados
```

```
##         V1       V2       V3       V4       V5
## 1 27.77448 28.06686 24.40896 21.41979 21.79201
## 2 14.06179 20.17816 22.41102 19.19151 21.51215
## 3 20.75906 26.57479 24.82876 29.67786 18.02943
## 4 14.56934 24.89084 15.92715 28.61615 23.94070
```

```
dados[1,3] # dado da linha 1 e coluna 3
```

```
## [1] 24.40896
```

```
dados[2,]  # dados da linha 2
```

```
##         V1       V2       V3       V4       V5
## 2 14.06179 20.17816 22.41102 19.19151 21.51215
```

```
dados[,5]  # dados da coluna 5
```

```
## [1] 21.79201 21.51215 18.02943 23.94070
```

```
dados[2:3,2:4] # submatriz com dados das linhas 2 e 3 e colunas 2 a 4
```

```
##         V2       V3       V4
## 2 20.17816 22.41102 19.19151
## 3 26.57479 24.82876 29.67786
```

Segue outras funções úteis. teste-as e verifique o que acontece. Leia também a ajuda desta funções:

- ls()
- dir()
- getwd()

A sequência da apredizagem do R se dá com uso, leitura de manuais, ajuda e fóruns da internet... Mãos a obra

Atividades

1. Suponha que você marcou o tempo que leva para chegar a cada uma de suas parcelas no campo. Os tempos em minutos foram: 18, 14, 14, 15, 14, 34, 16, 17, 21, 26. Passe estes valores para o R, chame o objeto de tempo. Usando funções do R ache o tempo máximo, mínimo e o tempo médio que você levou gasta para chegar em suas parcelas.

- Ops, o valor 34 foi um erro, ele na verdade é 15. Sem digitar tudo novamente, e usando colchetes [ ], mude o valor e calcule novamente o tempo médio.

2. Calcule o módulo de $2^3 \times -3^2$
3. Suponha que você coletou 10 amostras em duas reservas, as 5 primeiras amostras foram na reserva A e as 5 ultimas na reserva B. Use a função **rep** para criar um objeto chamado locais que contenha 5 letras A seguidas por cinco letras B.
4. Suponha que você deseje jogar na mega-sena, mas não sabe quais números jogar, use a função **sample** do R para escolher seis números para você. Lembre que a mega-sena tem valores de 1 a 60.
5. Einstein disse que Deus não joga dados, mas o R joga! Simule o resultado de 25 jogadas de um dado.
6. Crie um objeto com estes dados: 9 0 10 13 15 17 18 17 22 11 15 e chame-o de temp. Agora faça as seguintes transformações com esses dados:

- raiz quadrada de temp,
- log natural de temp,
- log(x+1) de temp,
- eleve os valores de temp ao quadrado.

7. Crie um objeto chamado info que contem seu nome, idade, altura, peso, email e telefone.

Análise de Clusters

Análise de Clusters ou *Cluster Analysis* tem por objetivo determinar padrões de variáveis que categorizem grupos, isto é, que identifiquem grupos de indivíduos.

Quanto ao posicionamento, assim como nos modelos de análise fatorial, a análise de clusters prevê no modelo apenas entradas (variáveis independentes) métricas.

Difere-se da análise discriminante (ou regressão Logística, por ex.) uma vez que nestas, eu sei a qual grupo os elementos da amostra pertencem.

Na análise de cluster o agrupamento dos elementos da amostra em conglomerados se dá pelas similaridades nas respostas das k variáveis independentes utilizadas na análise.

A expressão *Cluster analysis* foi utilizada pela primeira vez por Tryon, em 1939.

Características

- Detecta grupos homogêneos nos dados
- Grupamento de indivíduos ou objetos em grupos desconhecidos
- Não faz distinção entre variáveis dependentes e independentes

Observações

- A análise de aglomerados pode ser caracterizada como uma análise descritiva, não teórica e não inferencial.
- É utilizada principalmente com uma técnica exploratória.

Suposições

- Amostra deve ser representativa da população
- Multicolinearidade mínima
  - Devo evitar que duas ou mais variáveis repliquem o mesmo comportamento entre os grupos
  - Analisar correlações ($> 0{,}8$ pode representar um multicolinearidade)
- Amostra deve estar livre de *outliers*: Consulta visual via diagrama de perfis ou box-plots
- Razão (núm casos / núm clusters) deve ser razoável

Entre os principais **Critérios de medição da distância** estão:

- Distância Euclidiana: $D(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
- Distância Euclidiana ao quadrado
- Distância de Manhattan: $D(x, y) = \sum_{i=1}^{n} |x_i - y_i|$
- entre outras...

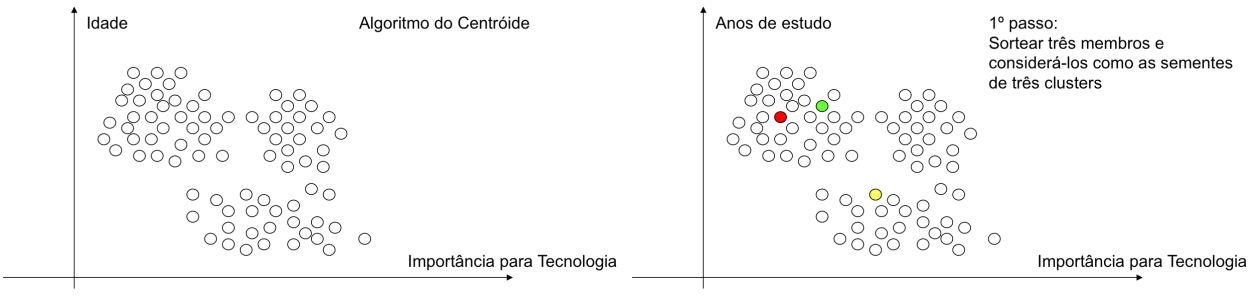
O R tem uma vasta variedade de funções para Análise de Clusters. Na aula de hoje nós vamos discutir uma das principais abordagens: Partitioning ou K-menas ou K-médias. O algoritmo k-means é um algoritmo que utiliza o cálculo de kcentróides para o agrupamento dos k clusters. Embora não haja uma solução ótima para o problema de determinar o número de clusters a serem extraídos, uma abordagem promissora é discutida na aula de hoje.

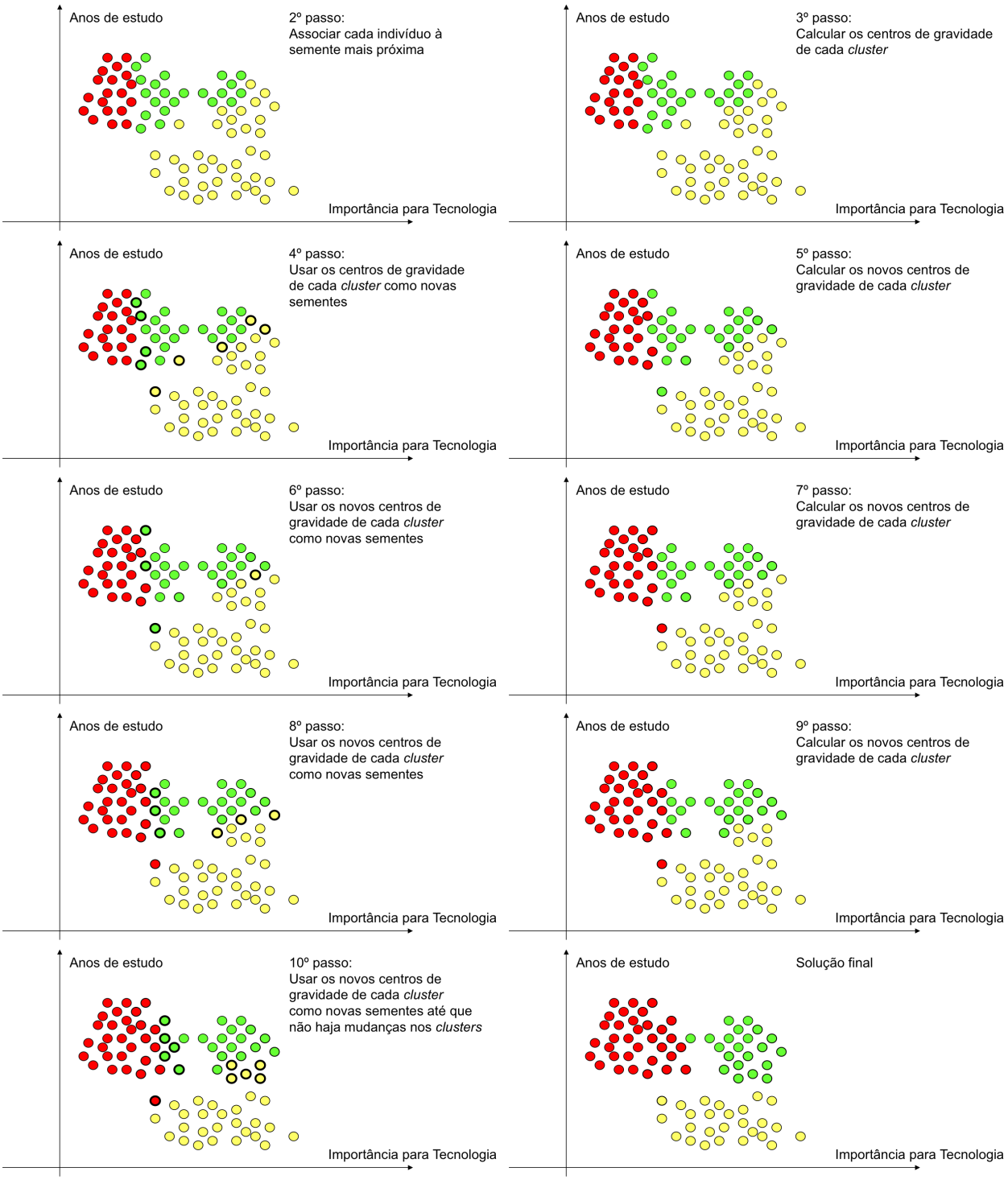
K-means

O cluster K-means é o método de particionamento mais popular. Requer que o analista especifique o número de clusters a serem extraídos. Uma plotagem da soma dos quadrados dentro dos grupos pelo número de clusters extraídos pode ajudar a determinar o número apropriado de clusters.

Note que a aplicação do método traz algumas exigências como o fato de lidar apenas com variáveis quantitativas (preferêncialmente contínuas), não haver dados faltantes (ver a função **na.omit**) e os dados serem inicialmente padronizados (ver a função **scale**).

A sequencia de imagens a seguir exemplifica o funcionamento do método para onde deseja-se agrupar os respondentes pelo número de anos de estudo e importância para a tecnologia.



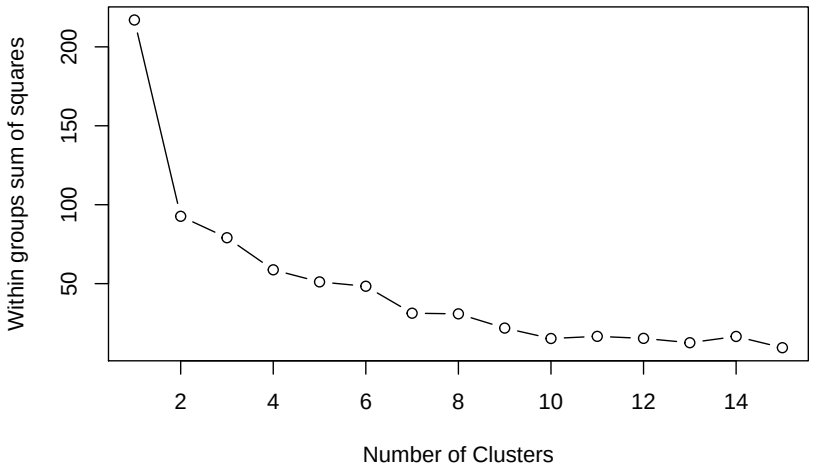


Considere inicialmente a base de dados **mtcars**. Vamos isolar as variáveis quantitativas de interesse, excluir os casos com dados faltantes e padronizar as variáveis. Na sequência, vamos construir um gráfico para escolher o número de agrupamentos.

```
require(car)
attach(mtcars)
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
cluster<-na.omit(mtcars)
cluster<-scale(cluster)
.cluster <- cluster[,1:7] # Apenas variáveis quantitativas
# Determine number of clusters
wss <- (nrow(.cluster)-1)*sum(apply(.cluster,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(.cluster, centers=i)$withinss)
plot(1:15, wss, type="b", xlab="Number of Clusters",
ylab="Within groups sum of squares")
```



A análise do gráfico é similar a análise do scree-plot, que veremos mais detalhadamente na aula de Análise fatorial. Por enquanto, basta saber que devemos escolher o ponto de corte que indica o número de clusters quando as inclinações do gráfico tendem a ser iguais a zero.

Uma vez definido o número clusters, o R possui funções para a identfiicação destes. Execute o conjunto de comandos a seguir:

```
# K-Means Cluster Analysis
fit <- kmeans(.cluster, 6) # 6 cluster solution
# get cluster means
aggregate(.cluster,by=list(fit$cluster),FUN=mean)
```

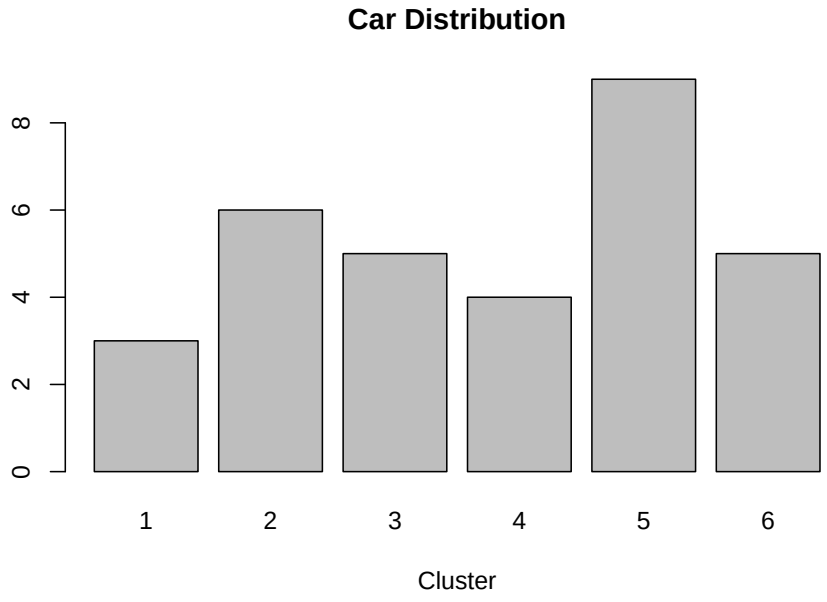
```
##   Group.1         mpg        cyl       disp         hp       drat         wt
## 1       1 -1.37006186  1.0148821  1.8284157  1.0206569 -1.0159917  2.1691456
## 2       2  1.65523937 -1.2248578 -1.1624447 -1.0382807  1.2252295 -1.3738462
## 3       3  0.41304073 -1.2248578 -0.8342986 -0.8092937  0.4814807 -0.4458549
## 4       4 -0.91101250  1.0148821  0.8857454  1.8313484  0.1467002  0.3273009
## 5       5 -0.43905813  0.7660222  0.5521575  0.1374448 -1.0762564  0.4019649
## 6       6 -0.05817621 -0.1049878 -0.5702971 -0.2696430  0.4777402 -0.1923947
##          qsec
```

```
## 1 -0.06085812
## 2  0.30755500
## 3  1.21730224
## 4 -1.54523655
## 5  0.01624438
## 6 -0.34290401
```

```
# append cluster assignment
.cluster <- data.frame(.cluster, fit$cluster)
head(.cluster)
```

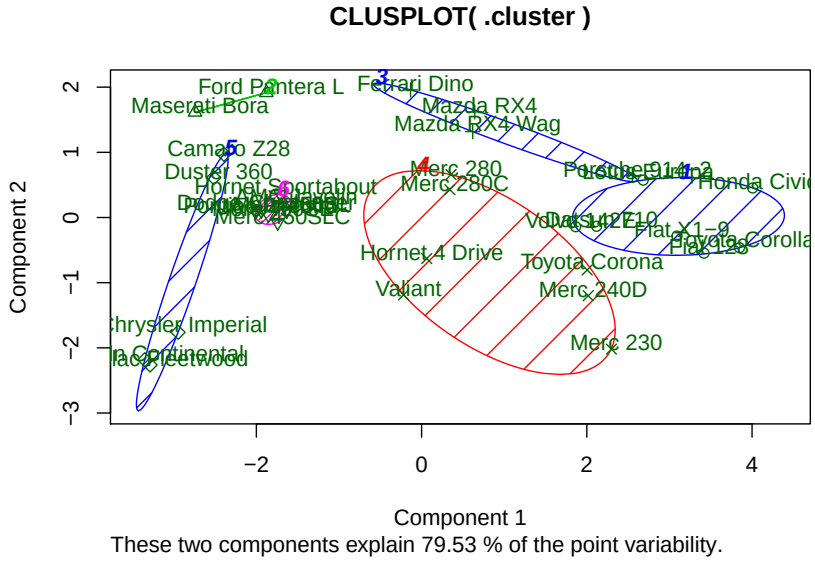
```
##                          mpg        cyl        disp         hp       drat
## Mazda RX4          0.1508848 -0.1049878 -0.57061982 -0.5350928  0.5675137
## Mazda RX4 Wag      0.1508848 -0.1049878 -0.57061982 -0.5350928  0.5675137
## Datsun 710         0.4495434 -1.2248578 -0.99018209 -0.7830405  0.4739996
## Hornet 4 Drive     0.2172534 -0.1049878  0.22009369 -0.5350928 -0.9661175
## Hornet Sportabout -0.2307345  1.0148821  1.04308123  0.4129422 -0.8351978
## Valiant           -0.3302874 -0.1049878 -0.04616698 -0.6080186 -1.5646078
##                           wt       qsec fit.cluster
## Mazda RX4         -0.610399567 -0.7771651           6
## Mazda RX4 Wag     -0.349785269 -0.4637808           6
## Datsun 710        -0.917004624  0.4260068           3
## Hornet 4 Drive    -0.002299538  0.8904872           5
## Hornet Sportabout  0.227654255 -0.4637808           5
## Valiant            0.248094592  1.3269868           5
```

Um diagrama de colunas pode mostrar a distribuição dos casos entre os clusters. **Execute um código apropriado para construir um gráfico similar ao apresentado a seguir**.

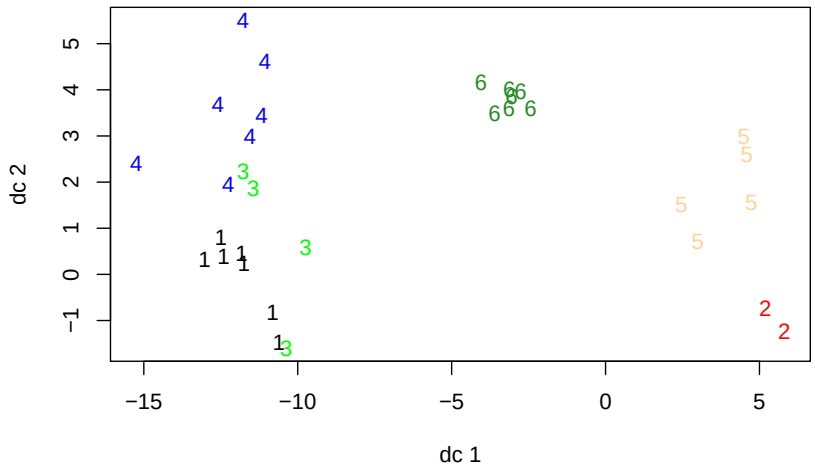


É sempre uma boa idéia analisar visualmente os resultados dos clusters...

```
# K-Means Clustering with 6 clusters
fit <- kmeans(cluster, 6)

# Cluster Plot against 1st 2 principal components

# vary parameters for most readable graph
library(cluster)
clusplot(.cluster, fit$cluster, color=TRUE, shade=TRUE,
   labels=2, lines=0)

# Centroid Plot against 1st 2 discriminant functions
library(fpc)
```




```
plotcluster(.cluster, fit$cluster)
```



Aula 05: Análise Fatorial Exploratória

Quando falamos de análises onde buscamos estabelecer relações de causa e efeito ou que interferem uma no comportamento da outra, consideramos ao menos uma variável independente que tenta explicar o comportamento de uma variável dependente. Em análise multivariada, tem-se um conjunto de variáveis independentes e deseja-se a construção de índices ou indicadores que representem fatores ocultos. Estes fatores chamam-se variáveis latentes e usualmente representam constructos teóricos que norteiam a escolha das variáveis escolhidas.

O objetivo da análise fatorial é o redimensionamento da base de dados de um número v de variáveis mensuráveis para um número k ($k << v$) de variáveis latentes. Em geral são necessárias grandes amostras para a execução destes procedimentos. A literatura recomenda pelo menos 10 casos por variável mensurada.

O procedimento possui pressupostos básicos que necessitam ser verificados:

- Variáveis correlacionadas ($r > 0,30$);
- Teste de esfericidade de Bartlett significativo (verifica se a matriz de correlações difere de uma matriz identidade);
- Adequação da amostra (KMO > 0,60).

Existem vários métodos de extração de fatores, sendo os mais usuais

- *Principal componentes*: Avalia a variância total;
- *Axis factoring*: Avalia apenas a variância comum através das comunalidades (medida da variância compartilhada das variáveis)

A retenção dos fatores pode ser por diferentes regras: Autovalor > 1, número de fatores pré-definido, percentagem de variância explicada pré-definida e Scree-plot são os mais usuais. A fim de melhorar a interpretação dos fatores, pode-se aplicar uma rotação na solução encontrada:

- Quatimax: cria um super fator inicial
- Varimax: distribui as importâncias dos fatores e concentra as cargas das variáveis
- Equimax: combinação dos anteriores
- Oblíquos: geram rotações sem respeitar a ortogonalidade dos fatores

A cada variável é definido um peso "de participação" em cada fator. As variáveis com maiores pesos ditam o significado do fator. Em geral deseja-se cargas fatoriais > 0,4. A tabela apresenta os limites de significância estatística para as cargas fatorias em função do tamanho da amostra com um poder de 80% e nível de significância de 5%.

Carga Fatorial	Tamanho Mínimo de Amostra
0,30	350
0,35	250
0,40	200
0,45	150
0,50	120
0,55	100
0,60	85
0,65	70
0,70	60
0,75	50

A questão fundamental da análise é

- Atribuir um nome descritivo aos fatores retidos
- Detectar a essência das variáveis individuais
- Abstrair o objeto de pesquisa

Componentes Principais

A função **princomp( )** produz uma análise de componentes principais sem rotação. Considere a base de dados *USArrests*.

```
# Principal Components Analysis
# entering raw data and extracting PCs
# from the correlation matrix
attach(USArrests)
head(USArrests)
```

```
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
```

```
mydata <- USArrests
summary(mydata)
```

```
##      Murder          Assault         UrbanPop          Rape
##  Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   : 7.30
##  1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st Qu.:15.07
##  Median : 7.250   Median :159.0   Median :66.00   Median :20.10
##  Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean   :21.23
##  3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd Qu.:26.18
##  Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.   :46.00
```

```
fit <- princomp(mydata, cor=TRUE)
summary(fit) # print variance accounted for
```

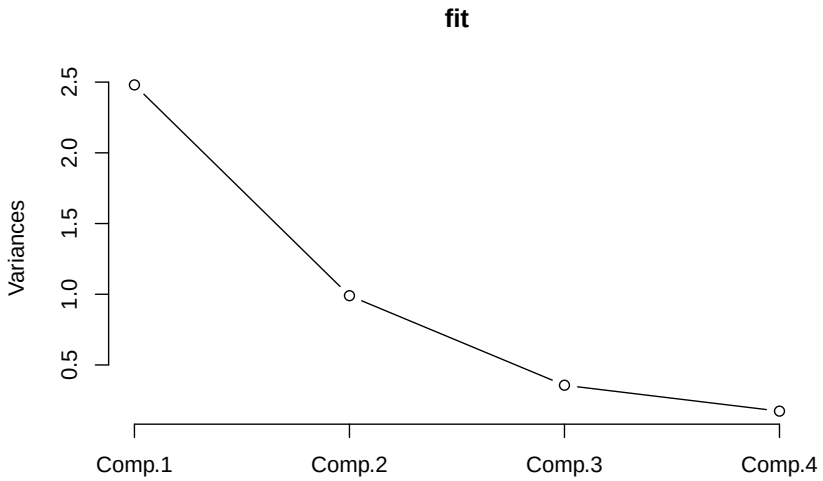
```
## Importance of components:
##                           Comp.1    Comp.2    Comp.3     Comp.4
## Standard deviation     1.5748783 0.9948694 0.5971291 0.41644938
## Proportion of Variance 0.6200604 0.2474413 0.0891408 0.04335752
## Cumulative Proportion  0.6200604 0.8675017 0.9566425 1.00000000
```

```
loadings(fit) # pc loadings
```

```
##
## Loadings:
##          Comp.1 Comp.2 Comp.3 Comp.4
## Murder    0.536  0.418  0.341  0.649
## Assault   0.583  0.188  0.268 -0.743
```

```
## UrbanPop  0.278 -0.873  0.378  0.134
## Rape      0.543 -0.167 -0.818
##
##                Comp.1 Comp.2 Comp.3 Comp.4
## SS loadings      1.00   1.00   1.00   1.00
## Proportion Var   0.25   0.25   0.25   0.25
## Cumulative Var   0.25   0.50   0.75   1.00
```

```
plot(fit,type="lines") # scree plot
```

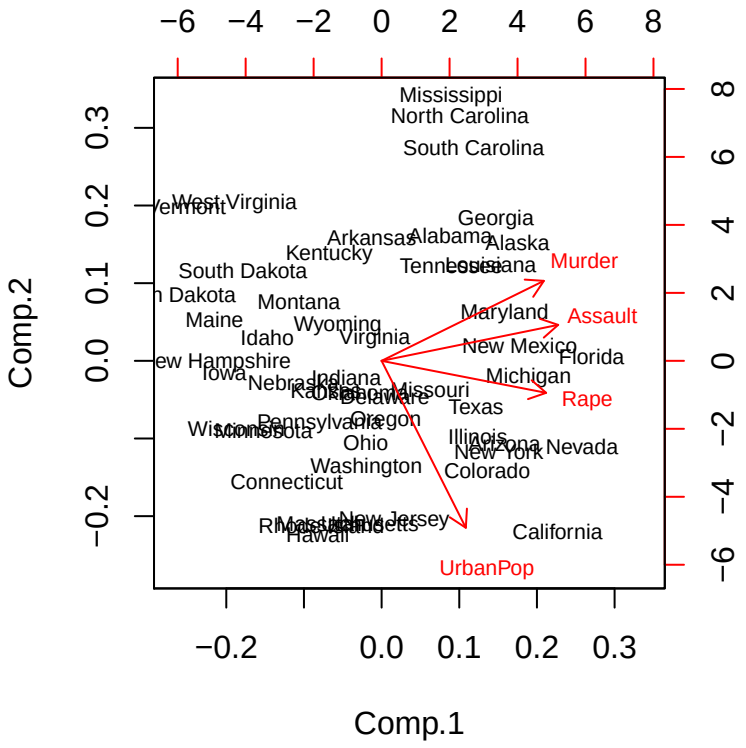



```
fit$scores # the principal components
```

```
##                    Comp.1      Comp.2      Comp.3       Comp.4
## Alabama         0.98556588  1.13339238  0.44426879  0.156267145
## Alaska          1.95013775  1.07321326 -2.04000333 -0.438583440
## Arizona         1.76316354 -0.74595678 -0.05478082 -0.834652924
## Arkansas       -0.14142029  1.11979678 -0.11457369 -0.182810896
## California      2.52398013 -1.54293399 -0.59855680 -0.341996478
## Colorado        1.51456286 -0.98755509 -1.09500699  0.001464887
## Connecticut    -1.35864746 -1.08892789  0.64325757 -0.118469414
## Delaware        0.04770931 -0.32535892  0.71863294 -0.881977637
## Florida         3.01304227  0.03922851  0.57682949 -0.096284752
## Georgia         1.63928304  1.27894240  0.34246008  1.076796812
## Hawaii         -0.91265715 -1.57046001 -0.05078189  0.902806864
## Idaho          -1.63979985  0.21097292 -0.25980134 -0.499104101
## Illinois        1.37891072 -0.68184119  0.67749564 -0.122021292
## Indiana        -0.50546136 -0.15156254 -0.22805484  0.424665700
## Iowa           -2.25364607 -0.10405407 -0.16456432  0.017555916
## Kansas         -0.79688112 -0.27016470 -0.02555331  0.206496428
## Kentucky       -0.75085907  0.95844029  0.02836942  0.670556671
## Louisiana       1.56481798  0.87105466  0.78348036  0.454728038
## Maine          -2.39682949  0.37639158  0.06568239 -0.330459817
## Maryland        1.76336939  0.42765519  0.15725013 -0.559069521
## Massachusetts  -0.48616629 -1.47449650  0.60949748 -0.179598963
## Michigan        2.10844115 -0.15539682 -0.38486858  0.102372019
## Minnesota      -1.69268181 -0.63226125 -0.15307043  0.067316885
## Mississippi     0.99649446  2.39379599  0.74080840  0.215508013
```

```
## Missouri        0.69678733 -0.26335479 -0.37744383  0.225824461
## Montana        -1.18545191  0.53687437 -0.24688932  0.123742227
## Nebraska       -1.26563654 -0.19395373 -0.17557391  0.015892888
## Nevada          2.87439454 -0.77560020 -1.16338049  0.314515476
## New Hampshire  -2.38391541 -0.01808229 -0.03685539 -0.033137338
## New Jersey      0.18156611 -1.44950571  0.76445355  0.243382700
## New Mexico      1.98002375  0.14284878 -0.18369218 -0.339533597
## New York        1.68257738 -0.82318414  0.64307509 -0.013484369
## North Carolina  1.12337861  2.22800338  0.86357179 -0.954381667
## North Dakota   -2.99222562  0.59911882 -0.30127728 -0.253987327
## Ohio           -0.22596542 -0.74223824  0.03113912  0.473915911
## Oklahoma       -0.31178286 -0.28785421  0.01530979  0.010332321
## Oregon          0.05912208 -0.54141145 -0.93983298 -0.237780688
## Pennsylvania   -0.88841582 -0.57110035  0.40062871  0.359061124
## Rhode Island   -0.86377206 -1.49197842  1.36994570 -0.613569430
## South Carolina  1.32072380  1.93340466  0.30053779 -0.131466685
## South Dakota   -1.98777484  0.82334324 -0.38929333 -0.109571764
## Tennessee       0.99974168  0.86025130 -0.18808295  0.652864291
## Texas           1.35513821 -0.41248082  0.49206886  0.643195491
## Utah           -0.55056526 -1.47150461 -0.29372804 -0.082314047
## Vermont        -2.80141174  1.40228806 -0.84126309 -0.144889914
## Virginia       -0.09633491  0.19973529 -0.01171254  0.211370813
## Washington     -0.21690338 -0.97012418 -0.62487094 -0.220847793
## West Virginia  -2.10858541  1.42484670 -0.10477467  0.131908831
## Wisconsin      -2.07971417 -0.61126862  0.13886500  0.184103743
## Wyoming        -0.62942666  0.32101297  0.24065923 -0.166651801
```

```
biplot(fit, cex=0.65)
```



Use **cor=FALSE** para que as componetes principais sejam extraídas a partir da matriz de covariâncias.

Use a opção **covmat=** para entrar com a matriz de correlação ou covariância diretamente. Se entrar com a matriz de covariâncias, a opção **n.obs=** é obrigatória.

A função **principal( )** do pacote **psych** pode ser utilizada para extrair e rotar os fatores pelo método das componentes principais.

```
# Varimax Rotated Principal Components
# retaining 5 components
library(psych)
fit <- principal(mydata, nfactors=4, rotate="varimax")
fit # print results
```

```
## Principal Components Analysis
## Call: principal(r = mydata, nfactors = 4, rotate = "varimax")
## Standardized loadings (pattern matrix) based upon correlation matrix
##           RC1   RC2  RC3  RC4 h2       u2 com
## Murder   0.91 -0.01 0.26 0.31  1  3.3e-16 1.4
## Assault  0.52  0.13 0.33 0.78  1 -2.2e-16 2.2
## UrbanPop 0.01  0.98 0.18 0.08  1  3.0e-15 1.1
## Rape     0.28  0.23 0.90 0.25  1  1.2e-15 1.5
##
##                        RC1  RC2  RC3  RC4
## SS loadings           1.18 1.03 1.02 0.77
## Proportion Var        0.30 0.26 0.25 0.19
## Cumulative Var        0.30 0.55 0.81 1.00
## Proportion Explained  0.30 0.26 0.25 0.19
## Cumulative Proportion 0.30 0.55 0.81 1.00
##
## Mean item complexity =  1.6
## Test of the hypothesis that 4 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0
##  with the empirical chi square  0  with prob <  NA
##
## Fit based upon off diagonal values = 1
```

mydata pode ser a matriz de dados brutos ou a matriz de covariâncias. Exclusão pareada (*Pairwise deletion*) dos dados faltantes é utilizado e o tipo de rotação pode ser *none*, *varimax*, *quatimax*, *promax*, *oblimin*, *simplimax*, ou *cluster*.

Análise Fatorial Exploratória

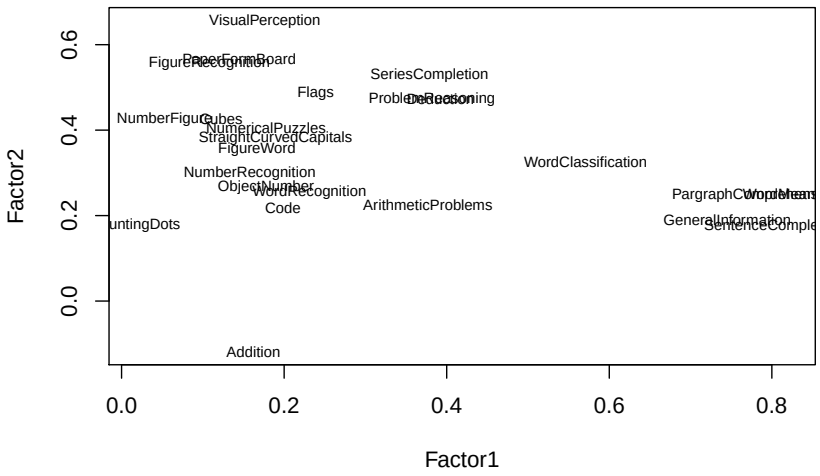
A função **factanal( )** produz a análise fatorial pela função de máxima verossimilhança. Vamos considerar a matriz de correlação de 24 testes psicológicos aplicados em 145 alunos desétima e oitava séries num suburbio de Chicago por Holzinger e Swineford.

```
attach(Harman74.cor)
mydata <- Harman74.cor$cov
# Maximum Likelihood Factor Analysis
# entering raw data and extracting 3 factors,
# with varimax rotation
fit <- factanal(factors = 3, covmat = Harman74.cor, rotation = "varimax")
print(fit, digits=2, cutoff=.3, sort=TRUE)
```

```
##
## Call:
## factanal(factors = 3, covmat = Harman74.cor, rotation = "varimax")
##
## Uniquenesses:
##       VisualPerception                  Cubes         PaperFormBoard
##                   0.50                   0.79                   0.66
##                  Flags     GeneralInformation  PargraphComprehension
##                   0.69                   0.35                   0.32
##     SentenceCompletion     WordClassification            WordMeaning
##                   0.30                   0.50                   0.26
##               Addition                   Code           CountingDots
##                   0.20                   0.59                   0.49
## StraightCurvedCapitals        WordRecognition      NumberRecognition
##                   0.57                   0.84                   0.85
##      FigureRecognition           ObjectNumber           NumberFigure
##                   0.64                   0.78                   0.64
##             FigureWord              Deduction       NumericalPuzzles
##                   0.79                   0.59                   0.58
##       ProblemReasoning       SeriesCompletion     ArithmeticProblems
##                   0.60                   0.50                   0.50
##
## Loadings:
##                         Factor1 Factor2 Factor3
## GeneralInformation       0.75
## PargraphComprehension    0.78
## SentenceCompletion       0.80
## WordClassification       0.57    0.33
## WordMeaning              0.82
## VisualPerception                 0.66
## PaperFormBoard                   0.56
## FigureRecognition                0.56
## SeriesCompletion         0.38    0.53
## Addition                                 0.87
## Code                                     0.57
## CountingDots                             0.69
## ArithmeticProblems       0.38            0.55
## Cubes                            0.43
## Flags                            0.49
## StraightCurvedCapitals           0.38    0.50
## WordRecognition
## NumberRecognition
## ObjectNumber                             0.34
## NumberFigure                     0.43    0.42
## FigureWord                       0.35
## Deduction                0.39    0.47
## NumericalPuzzles                 0.41    0.47
## ProblemReasoning         0.38    0.47
##
##                Factor1 Factor2 Factor3
## SS loadings       3.80    3.49    3.19
## Proportion Var    0.16    0.15    0.13
## Cumulative Var    0.16    0.30    0.44
##
```

```
## Test of the hypothesis that 3 factors are sufficient.
## The chi square statistic is 295.59 on 207 degrees of freedom.
## The p-value is 5.12e-05
```

```
# plot factor 1 by factor 2
load <- fit$loadings[,1:2]
plot(load, type="n") # set up plot
text(load[,1], load[,2], labels=rownames(load), cex=0.7) # add variable names
```



A opção **rotation=** inclui as opções *varimax*, *promax*, e *none*. A opção **scores=** *regression* ou *Bartlett* determina como serão produzidos os escores. Use a opção **covmat=** para entrar com a matriz de correlação ou covariância diretamente. Se entrar com a matriz de covariâncias, a opção **n.obs=** é obrigatória.

A função **fa( )** do pacote **psych** ofecere uma série de funções relacioandas a análise fatorial, incluindo *principal axis factoring*.

```
# Principal Axis Factor Analysis
library(psych)
fit <- fa(mydata, nfactors=3, rotate="varimax")
fit # print results
```

```
## Factor Analysis using method =  minres
## Call: fa(r = mydata, nfactors = 3, rotate = "varimax")
## Standardized loadings (pattern matrix) based upon correlation matrix
##                          MR1   MR3  MR2   h2   u2 com
## VisualPerception        0.17  0.65 0.21 0.49 0.51 1.4
## Cubes                   0.12  0.43 0.09 0.21 0.79 1.2
## PaperFormBoard          0.16  0.55 0.01 0.33 0.67 1.2
## Flags                   0.24  0.49 0.09 0.31 0.69 1.5
## GeneralInformation      0.74  0.19 0.24 0.64 0.36 1.4
## PargraphComprehension   0.77  0.25 0.14 0.67 0.33 1.3
## SentenceCompletion      0.82  0.17 0.16 0.72 0.28 1.2
## WordClassification      0.57  0.33 0.25 0.50 0.50 2.0
## WordMeaning             0.82  0.24 0.11 0.74 0.26 1.2
## Addition                0.17 -0.13 0.83 0.74 0.26 1.1
## Code                    0.18  0.18 0.63 0.46 0.54 1.3
## CountingDots            0.03  0.15 0.67 0.47 0.53 1.1
## StraightCurvedCapitals 0.19  0.35 0.51 0.42 0.58 2.1
```

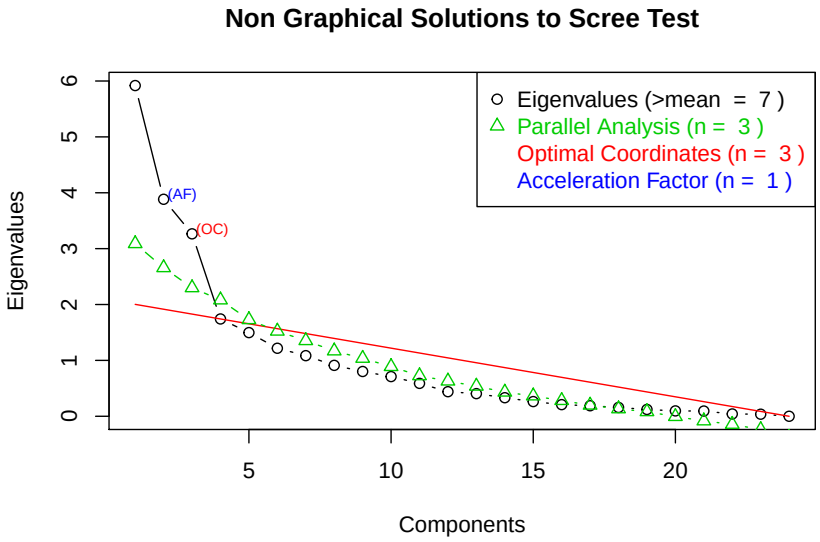
```
## WordRecognition        0.22  0.24 0.26 0.17 0.83 2.9
## NumberRecognition      0.14  0.29 0.25 0.16 0.84 2.4
## FigureRecognition      0.10  0.56 0.22 0.37 0.63 1.4
## ObjectNumber           0.16  0.25 0.39 0.24 0.76 2.1
## NumberFigure           0.04  0.42 0.47 0.39 0.61 2.0
## FigureWord             0.16  0.34 0.28 0.22 0.78 2.4
## Deduction              0.39  0.48 0.18 0.42 0.58 2.2
## NumericalPuzzles       0.19  0.40 0.46 0.41 0.59 2.3
## ProblemReasoning       0.38  0.46 0.21 0.40 0.60 2.4
## SeriesCompletion       0.39  0.53 0.27 0.50 0.50 2.4
## ArithmeticProblems     0.37  0.22 0.55 0.49 0.51 2.1
##
##                         MR1  MR3  MR2
## SS loadings            3.76 3.37 3.34
## Proportion Var         0.16 0.14 0.14
## Cumulative Var         0.16 0.30 0.44
## Proportion Explained   0.36 0.32 0.32
## Cumulative Proportion  0.36 0.68 1.00
##
## Mean item complexity =  1.8
## Test of the hypothesis that 3 factors are sufficient.
##
## The degrees of freedom for the null model are  276  and the objective function was  11.44
## The degrees of freedom for the model are 207  and the objective function was  2.24
##
## The root mean square of the residuals (RMSR) is  0.05
## The df corrected root mean square of the residuals is  0.06
##
## Fit based upon off diagonal values = 0.97
## Measures of factor score adequacy
##                                                  MR1  MR3  MR2
## Correlation of (regression) scores with factors   0.93 0.88 0.92
## Multiple R square of scores with factors          0.87 0.78 0.84
## Minimum correlation of possible factor scores     0.74 0.56 0.69
```

mydata pode ser a matriz de dados brutos ou a matriz de covariâncias. Exclusão pareada (*Pairwise deletion*) dos dados faltantes é utilizado e o tipo de rotação pode ser *varimax* ou *promax*.

Determinando o número de Fatores para extrair

Uma decisão crucial na análise fatorial exploratória refere-se a quantos fatores eu devo reter. O pacote **nFactors** oferece um conjunto de funções para auxiliar nesta decisão. Vejamos o *Scree plot*.

```
# Determine Number of Factors to Extract
library(nFactors)
ev <- eigen(cor(mydata)) # get eigenvalues
ap <- parallel(subject=nrow(mydata),var=ncol(mydata),
  rep=100,cent=.05)
nS <- nScree(x=ev$values, aparallel=ap$eigen$qevpea)
plotnScree(nS)
```



Testes usuais de adequação da amostra

Os testes mais usuais para a análise fatorial exploratória são o teste de esfericidade de Barttlet, O teste de K-M-O e a medida de consistência interna Alpha de Crombach. Os três procedimentos são apresentados a seguir e estão disponíveis no pacote **psych** do R.

Bartlett's Test

O teste de Bartlett verifica que a matriz de correlação não é uma matriz identidade. Uma vez que deseja-se criar variáveis latentes (fatores) a partir da combinação de variáveis que meçam o mesmo constructo, é de se esperar que estas sejam correlacionadas. Caso isto ocorra, a matriz de correlações não será uma matriz identidade (completa de zeros fora da diagonal principal). O teste verifica os residuos da diferença entre a matriz de correlação e uma matriz identidade.

Como entrada deve ser informado a matriz de correlação dos dados e o tamanho da amostra.

```
library(psych)
cortest.bartlett(mydata, n = 145, diag=TRUE)
```

```
## $chisq
## [1] 1545.862
##
## $p.value
## [1] 2.399561e-175
##
## $df
## [1] 276
```

Kaiser, Meyer, Olkin Measure Of Sampling Adequacy

Henry Kaiser (1970) introduziu uma Medida de Adequação de Amostragem (MSA) de matrizes de dados analíticos de fatores. Kaiser e Rice (1974) então o modificaram. Isso é apenas uma função dos elementos quadrados da matriz imagem em comparação com os quadrados das correlações originais. O MSA geral e as estimativas para cada item são encontrados. O índice é conhecido como índice Kaiser-Meyer-Olkin (KMO).

Como entrada deve ser informado a matriz de correlação dos dados.

```
library(psych)
KMO(mydata)
```

```
## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = mydata)
## Overall MSA =  0.88
## MSA for each item =
##       VisualPerception                 Cubes         PaperFormBoard
##                   0.90                  0.84                   0.78
##                  Flags     GeneralInformation  PargraphComprehension
##                   0.85                  0.88                   0.89
##     SentenceCompletion     WordClassification            WordMeaning
##                   0.89                  0.92                   0.88
##               Addition                  Code           CountingDots
##                   0.81                  0.85                   0.84
## StraightCurvedCapitals        WordRecognition      NumberRecognition
##                   0.89                  0.85                   0.88
##      FigureRecognition          ObjectNumber           NumberFigure
##                   0.89                  0.85                   0.88
##             FigureWord             Deduction       NumericalPuzzles
##                   0.83                  0.93                   0.91
##       ProblemReasoning       SeriesCompletion     ArithmeticProblems
##                   0.93                  0.91                   0.92
```

Crombach's Alpha

Alpha é uma das várias estimativas da confiabilidade da consistência interna de um teste. Etenda aqui como teste um questionário ou instrumento de coleta de dados. Como resultado, o teste informa um escore que aponta a consistência interna das respostas obteidas por meio do instrumento de coleta (o teste). Deve-se imputar a base de dados brutos ou a matriz de correlação em conjunto com o tamanho da amostra.

```
library(psych)
alpha(mydata, n.obs = 145)
```

```
##
## Reliability analysis
## Call: alpha(x = mydata, n.obs = 145)
##
##   raw_alpha std.alpha G6(smc) average_r S/N   ase median_r
##       0.91      0.91    0.94       0.3  10 0.011     0.29
##
##  lower alpha upper     95% confidence boundaries
## 0.89 0.91 0.93
##
##  Reliability if an item is dropped:
##                       raw_alpha std.alpha G6(smc) average_r  S/N alpha se
## VisualPerception           0.91      0.91    0.93      0.30  9.8    0.011
## Cubes                      0.91      0.91    0.94      0.31 10.3    0.011
## PaperFormBoard             0.91      0.91    0.93      0.31 10.2    0.011
## Flags                      0.91      0.91    0.93      0.30 10.1    0.011
```

```
## GeneralInformation          0.91      0.91    0.93      0.30  9.7    0.011
## PargraphComprehension       0.91      0.91    0.93      0.30  9.7    0.011
## SentenceCompletion          0.91      0.91    0.93      0.30  9.7    0.011
## WordClassification          0.91      0.91    0.93      0.30  9.7    0.011
## WordMeaning                 0.91      0.91    0.93      0.30  9.7    0.011
## Addition                    0.91      0.91    0.93      0.31 10.2    0.011
## Code                        0.91      0.91    0.93      0.30  9.9    0.011
## CountingDots                0.91      0.91    0.93      0.31 10.1    0.011
## StraightCurvedCapitals      0.91      0.91    0.93      0.30  9.8    0.011
## WordRecognition             0.91      0.91    0.94      0.31 10.2    0.011
## NumberRecognition           0.91      0.91    0.94      0.31 10.3    0.011
## FigureRecognition           0.91      0.91    0.93      0.30 10.0    0.011
## ObjectNumber                0.91      0.91    0.93      0.30 10.1    0.011
## NumberFigure                0.91      0.91    0.93      0.30  9.9    0.011
## FigureWord                  0.91      0.91    0.93      0.31 10.1    0.011
## Deduction                   0.91      0.91    0.93      0.30  9.8    0.011
## NumericalPuzzles            0.91      0.91    0.93      0.30  9.8    0.011
## ProblemReasoning            0.91      0.91    0.93      0.30  9.8    0.011
## SeriesCompletion            0.91      0.91    0.93      0.29  9.6    0.011
## ArithmeticProblems          0.91      0.91    0.93      0.30  9.7    0.011
##                          var.r med.r
## VisualPerception         0.016  0.28
## Cubes                    0.015  0.30
## PaperFormBoard           0.015  0.30
## Flags                    0.016  0.29
## GeneralInformation       0.014  0.29
## PargraphComprehension    0.014  0.29
## SentenceCompletion       0.014  0.29
## WordClassification       0.015  0.29
## WordMeaning              0.014  0.30
## Addition                 0.014  0.30
## Code                     0.016  0.29
## CountingDots             0.015  0.30
## StraightCurvedCapitals   0.016  0.29
## WordRecognition          0.016  0.30
## NumberRecognition        0.016  0.30
## FigureRecognition        0.016  0.29
## ObjectNumber             0.016  0.30
## NumberFigure             0.016  0.29
## FigureWord               0.016  0.30
## Deduction                0.016  0.29
## NumericalPuzzles         0.016  0.28
## ProblemReasoning         0.016  0.29
## SeriesCompletion         0.016  0.28
## ArithmeticProblems       0.016  0.28
##
##  Item statistics
##                             r r.cor r.drop
## VisualPerception         0.62  0.60   0.57
## Cubes                    0.42  0.38   0.35
## PaperFormBoard           0.46  0.43   0.40
## Flags                    0.51  0.48   0.45
## GeneralInformation       0.67  0.67   0.63
## PargraphComprehension    0.67  0.66   0.62
```

```
## SentenceCompletion     0.65  0.64   0.60
## WordClassification     0.67  0.66   0.63
## WordMeaning            0.66  0.66   0.62
## Addition               0.48  0.47   0.42
## Code                   0.59  0.57   0.54
## CountingDots           0.50  0.48   0.44
## StraightCurvedCapitals 0.62  0.61   0.57
## WordRecognition        0.47  0.43   0.41
## NumberRecognition      0.44  0.40   0.38
## FigureRecognition      0.55  0.53   0.50
## ObjectNumber           0.51  0.48   0.45
## NumberFigure           0.57  0.54   0.51
## FigureWord             0.49  0.46   0.43
## Deduction              0.63  0.62   0.59
## NumericalPuzzles       0.62  0.61   0.58
## ProblemReasoning       0.63  0.61   0.58
## SeriesCompletion       0.70  0.69   0.66
## ArithmeticProblems     0.67  0.66   0.63
```

Atividade

A base de dados do MAPEM (arquivo salvo no formato do SPSS) foi obtida por meio de um projeto da PETROBRÁS que buscava avaliar o impacto ambiental da perfuração de poços de petróleo em alto mar. As coletas das amostras de solo ocorreram em períodos de tempo, afastadas por uma dada distância em certa direção a partir do ponto onde um poço de petróleo estava sendo perfurado. São observadas variáveis de identfiicação do local de coleta e contagem de animais da macro e micro fauna, composição do solo, químicos e sedimentos. Explore a base de dados para identfiicar os elementos citados.

1. Faça um esboço da localização dos pontos de coleta a partir das informações da base (tenha como origem do sistema o ponto de perfuração do poço).
2. Verifique a adequação do procedimento de análise fatorial utilizando os três testes apresentados no final deste módulo de estudo.
3. Proceda uma análise fatorial exploratória. Apresente os resultados da análise, identificando o perfil de cada fator retido. Descreva sumariamente os procedimentos e decisões realizados na análise.
4. Proceda a uma análise de Cluster: identifique o número de Clusters, execute o procedimento e caracterize os clusters obtidos a partir das variáveis disponíveis.
5. Ao final do arquivo, apresente o código utilizado para a realização de toda a análise.

Revisão: Conceitos de Testes de Hipóteses e Aplicações

Hipóteses

Em um teste de hipóteses, desejamos testar a hipótese de que, provavelmente, o valor do parâmetro suposto seja verdadeiro (ou não) para a população de onde foi extraída a amostra. Em termos gerais, uma hipótese é uma conjectura sobre algum fenômeno ou conjunto de fatos. Em estatística inferencial o termo hipótese tem um significado bastante especifico. É uma conjectura sobre um ou mais parâmetros populacionais. O teste de hipóteses envolve fazer inferências sobre a natureza da população com base nas observações de uma amostra extraída desta população.

Uma hipótese estatística é uma suposição ou afirmação que pode ou não ser verdadeira, relativa a uma ou mais populações. A veracidade ou falsidade de uma hipótese estatística nunca é conhecida com certeza, a

menos que, se examine toda a população, o que é impraticável na maior parte das situações. Desta forma, toma-se uma amostra aleatória da população de interesse e com base nesta amostra é estabelecido se a hipótese é provavelmente verdadeira ou provavelmente falsa.

Em estatística trabalha-se com dois tipos de hipótese. A hipótese nula é a hipótese de igualdade. Esta hipótese é denominada de hipótese de nulidade e é representada por H_0 (lê-se h zero). A hipótese nula é normalmente formulada com o objetivo de ser rejeitada. A rejeição da hipótese nula envolve a aceitação de outra hipótese denominada de alternativa (H_1). Esta hipótese é a definição operacional da hipótese de pesquisa que se deseja comprovar. A natureza do estudo vai definir como deve ser formulada a hipótese alternativa. Por exemplo, se o parâmetro a ser testado é representado por θ, então a hipótese nula seria: $H_0 : \theta = \theta_0$ e as hipóteses alternativas poderiam ser: $H_1 : \theta \neq \theta_0$; $H_1 : \theta > \theta_0$ ou $H_1 : \theta < \theta_0$. No primeiro caso, $H_1 : \theta \neq \theta_0$, diz-se que o teste é bilateral (ou bicaudal), se $H_1 : \theta > \theta_0$, diz-se que o teste é unilateral (ou unicaudal) à direita e se $H_1 : \theta < \theta_0$, então, diz-se que o teste é unilateral (ou unicaudal) à esquerda.

A lógica de um teste de hipóteses pode ser descrita pela Figura 1.

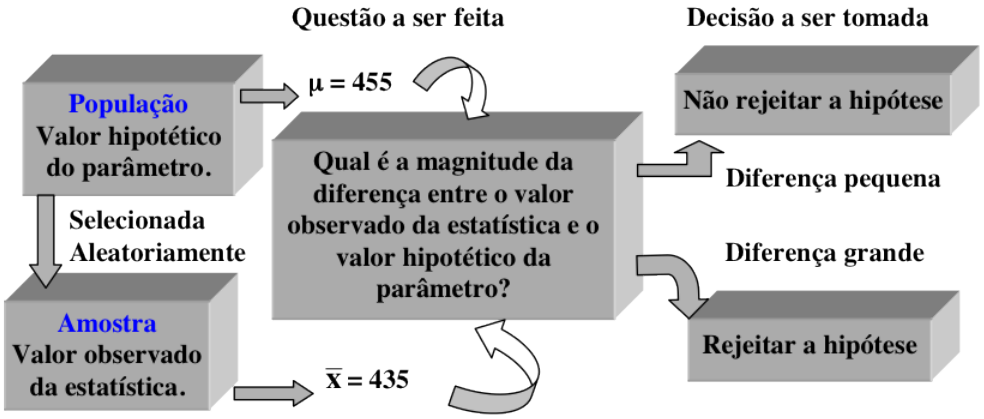


Figure 2: Lógica de um teste de hipóteses

Principais medidas e testes

Para as atividades a seguir, utilizaremos a base de dados XXXX disponível no ambiente virtual de aprendizagem. Carregue a base de dados no R antes de inciar. **Fique atendo ao nome atribuído a base!**

```
library(readxl)
mydata <- read_excel("Seguro_Residencial.xlsx", sheet = "Seguro")
```

Gerando tabelas de frequências

Você pode gerar tabelas de frequência usando a função **table( )**, tabelas de proporções usando a função **prop.table( )** e frequências marginais usando **margin.table( )**.

```
# 2-Way Frequency Table
mytable <- table(mydata$Tipo, mydata$Fraudulento)
# Tipo será linhas, Fraudulento será colunas
mytable # print table
```

```
##
##        0    1
##   1  963   91
##   2  577   50
##   3  919  120
##   4  378   26
##   5 1115  176
```

```
margin.table(mytable, 1) # Tipo frequencies (summed over Fraudulento)
```

```
##
##    1    2    3    4    5
## 1054  627 1039  404 1291
```

```
margin.table(mytable, 2) # Fraudulento frequencies (summed over Tipo)
```

```
##
##    0    1
## 3952  463
```

```
prop.table(mytable) # cell percentages
```

```
##
##               0           1
##   1 0.218120045 0.020611552
##   2 0.130690827 0.011325028
##   3 0.208154020 0.027180068
##   4 0.085617214 0.005889015
##   5 0.252548131 0.039864100
```

```
prop.table(mytable, 1) # row percentages
```

```
##
##              0          1
##   1 0.91366224 0.08633776
##   2 0.92025518 0.07974482
##   3 0.88450433 0.11549567
##   4 0.93564356 0.06435644
##   5 0.86367157 0.13632843
```

```
prop.table(mytable, 2) # column percentages
```

```
##
##              0          1
##   1 0.24367409 0.19654428
##   2 0.14600202 0.10799136
##   3 0.23254049 0.25917927
##   4 0.09564777 0.05615551
##   5 0.28213563 0.38012959
```

table( ) também pode gerar tabelas multidimensionais com base em 3 ou mais variáveis categóricas. Porém, a função **ftable( )** imprime os resultados de maneira mais atraente.

```
# 3-Way Frequency Table
mytable <- table(mydata$Tipo, mydata$Fraudulento, mydata$Formação)
ftable(mytable)
```

```
##        1   2   3   4   5
##
## 1 0  146 319 217 213  68
##   1   16  28  18  21   8
## 2 0  113 182 115 126  41
##   1    7  17  13   8   5
## 3 0  170 296 178 210  65
##   1   20  40  28  27   5
## 4 0   61 135  73  87  22
##   1    3   8   4   9   2
## 5 0  201 345 250 241  78
##   1   23  52  42  48  11
```

xtabs( ) permite criartabelas de contingência usando estilo de fórmula como *input*.

```
# 3-Way Frequency Table
mytable <- xtabs(~Tipo+Fraudulento+Formação, data=mydata)
ftable(mytable) # print table
```

```
##                  Formação   1   2   3   4   5
## Tipo Fraudulento
## 1    0                    146 319 217 213  68
##      1                     16  28  18  21   8
## 2    0                    113 182 115 126  41
##      1                      7  17  13   8   5
## 3    0                    170 296 178 210  65
##      1                     20  40  28  27   5
## 4    0                     61 135  73  87  22
##      1                      3   8   4   9   2
## 5    0                    201 345 250 241  78
##      1                     23  52  42  48  11
```

```
summary(mytable) # chi-square test of indepedence
```

```
## Call: xtabs(formula = ~Tipo + Fraudulento + Formação, data = mydata)
## Number of cases in table: 4415
## Number of factors: 3
## Test for independence of all factors:
##  Chisq = 55.71, df = 40, p-value = 0.05043
##  Chi-squared approximation may be incorrect
```

Se uma variável for incluída no lado esquerdo da fórmula, será considerado um vetor de frequências (útil se os dados já tiverem sido tabulados).

Testes de Independência

Teste Qui-Quadrado

Para tabelas bidirecionais, você pode usar **chisq.test(mytable)** para testar a independência da variável de linha e coluna. Por padrão, o valor p é calculado a partir da distribuição qui-quadrado assintótica da estatística de teste. Opcionalmente, o valor-p pode ser derivado via simultânea de Monte Carlo.

Teste exato de Fisher

fisher.test(x) fornece um teste exato de independência. **x** é uma tabela de contingência bidimensional em forma de matriz.

Teste de Mantel-Haenszel

Use a função **mantelhaen.test(x)** para executar um teste qui-quadrado de Cochran-Mantel-Haenszel da hipótese nula de que duas variáveis nominais são condicionalmente independentes em cada estrato, assumindo que não haja interação de três vias. **x** é uma tabela de contingência tridimensional, em que a última dimensão se refere aos estratos.

Modelos Loglineares

Você pode usar a função **loglm()* no pacote MASS para produzir modelos lineares de log. Por exemplo, vamos supor que temos uma tabela de contingência de três vias com base nas variáveis Tipo, Fraudulento e Formação.

```
require(MASS)
mytable <- xtabs (~ Tipo + Fraudulento + Formação, data=mydata)
```

Podemos realizar teste para verificar a independência mútua: Tipo, Fraudulento e Formação são independentes por pares?

```
loglm (~ Tipo + Fraudulento + Formação, mytable)
```

```
## Call:
## loglm(formula = ~Tipo + Fraudulento + Formação, data = mytable)
##
## Statistics:
##                       X^2 df   P(> X^2)
## Likelihood Ratio 56.31422 40 0.04507874
## Pearson          55.71169 40 0.05043473
```

Medidas de associação

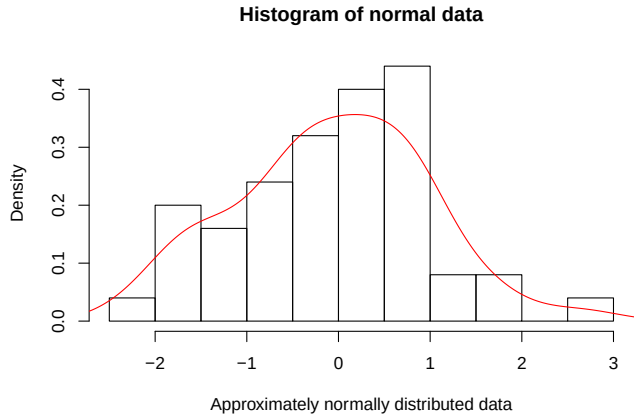
A função **assocstats(mytable)** no pacote **vcd** calcula o coeficiente phi, coeficiente de contingência e o Cramer's V para uma tabela r x c. A função **kappa(mytable)** no pacote **vcd** calcula Cohen's kappa e weighted kappa para uma matriz de confusão. Veja o artigo de Richard Darlington's Measures of Association in Crosstab Tables para uma revisão sobre estas estatísticas.

Testes de normalidade

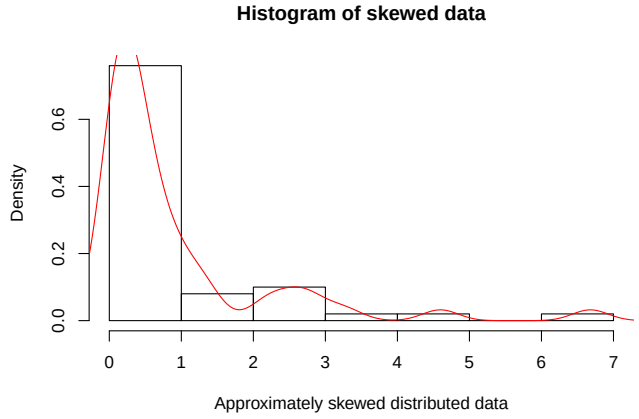
Uma das premissas para a maioria dos testes paramétricos serem confiáveis é que os dados são distribuídos aproximadamente como uma normal. A distribuição normal atinge o pico no meio e é simétrica em relação à média. Os dados não precisam ser perfeitamente distribuídos normalmente para que os testes sejam confiáveis, mas sua verificação é altamente recomendada para garantir os resultados em termos probabilísticos.

A plotagem de um histograma da variável de interesse fornecerá uma indicação da forma do distribuição. Uma curva de densidade suaviza o histograma e pode ser adicionada ao gráfico. Primeiro, produza o histograma para os dados normalmente distribuídos (normal) e adicione uma curva de densidade. Repita para os dados assimétricos (assim).

```
set.seed(2019)
normal <- rnorm(50)
assim <- rchisq(50,1)
hist(normal,probability=T, main="Histogram of normal data",
     xlab="Approximately normally distributed data")
lines(density(normal),col=2)
```



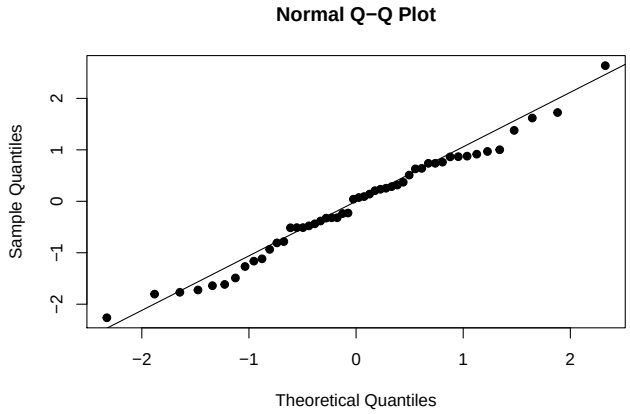

```
hist(assim,probability=T, main="Histogram of skewed data",
     xlab="Approximately skewed distributed data")
lines(density(assim),col=2)
```



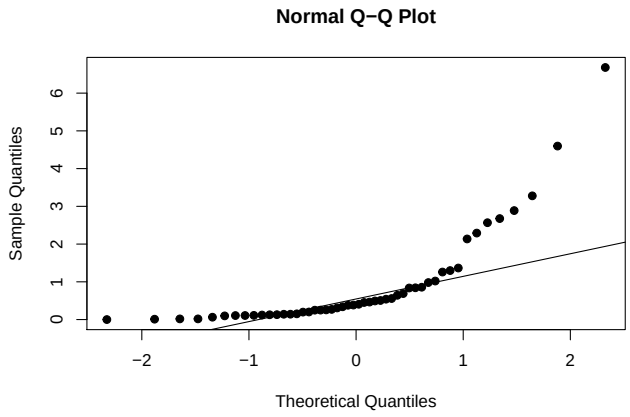
É muito improvável que um histograma dos dados da amostra produza uma curva normal perfeitamente suave, especialmente se o tamanho da amostra for pequeno. Desde que os dados sejam distribuídos aproximadamente normalmente, com um pico no meio e bastante simétrico, um teste paramétrico pode ser usado.

O gráfico Q-Q normal é um método gráfico alternativo de avaliar a normalidade com o histograma e é mais fácil de usar quando há amostras pequenas. A dispersão compara os dados com uma distribuição normal perfeita. A dispersão deve ficar o mais próximo possível da linha, sem que nenhum padrão óbvio saia da linha para que os dados sejam considerados normalmente distribuídos. Abaixo estão os mesmos exemplos de dados normalmente distribuídos e inclinados. Desenhe o gráfico qq dos dados normalmente distribuídos usando pch = 19 para produzir círculos sólidos. Após adicione uma linha em que x = y para ajudar a avaliar a proximidade com a dispersão. Repita para os dados assimétricos.

```
qqnorm (normal, principal = "gráfico QQ de dados normais", pch = 19)
qqline (normal)
```




```
qqnorm (assim, principal = "gráfico QQ de dados assimétricos", pch = 19)
qqline (assim)
```



Também existem métodos específicos para testar a normalidade, mas eles devem ser usados em conjunto com um histograma ou um gráfico Q-Q. O teste de Kolmogorov-Smirnov e o teste W de Shapiro-Wilk testam se a distribuição subjacente é normal. Ambos os testes são sensíveis a valores discrepantes e são influenciados pelo tamanho da amostra:

- Para amostras menores, é menos provável que a não normalidade seja detectada, mas o teste de Shapiro-Wilk deve ser preferido, pois geralmente é mais sensível.
- Para amostras maiores (ou seja, mais de cem), os testes de normalidade são excessivamente conservadores e a suposição de normalidade pode ser rejeitada com muita facilidade.
- Qualquer avaliação também deve incluir uma avaliação da normalidade de histogramas ou gráficos Q-Q e estes são mais apropriados para avaliar a normalidade em amostras maiores.

```r
shapiro.test(normal)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  normal
## W = 0.98406, p-value = 0.7306
```

```r
shapiro.test(assim)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  assim
## W = 0.66084, p-value = 1.749e-09
```

Existem ainda outras opções de testes de normalidade, como por exemplo a correção de Lilliefors para o teste de Kolmogorov-Smirnov, o Aderson darling test, ou ainda o Cramér-von-Misses test.

```r
# Executa a correção de Lilliefors para o teste de Kolmogorov-Smirnov (similar ao SPSS)
require(nortest)
lillie.test(normal)
```

```
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  normal
## D = 0.075711, p-value = 0.6734
```

```r
# Executa o teste de Anderson-Darling para a hipótese composta de normalidade
ad.test(normal)
```

```
##
##  Anderson-Darling normality test
##
## data:  normal
## A = 0.29239, p-value = 0.5911
```

```r
# Cramer-Von Mises Test For Normality
cvm.test(normal)
```

```
##
##  Cramer-von Mises normality test
##
## data:  normal
## W = 0.041142, p-value = 0.6531
```

Correlações

Pode-se usar a função **cor( )** para produzir correlações e a função **cov( )** para produzir as covariâncias.

O formato simplificado é **cor(x, use=, method=)** onde **x** é uma matriz com dados quantitativos, **use** especifica como lidar com os dados faltantes (missingata) - As opções são **all.obs** (assume que não existem dados faltantes), **complete.obs** (exclui toda a linha), e **pairwise.complete.obs** (exclui o par).

Infelizmente, nem **cor( )** ou **cov( )** produzem os testes de significância, mas pode-se utilizar a função **cor.test( )** para testar um simples coeficiente de correlação. A função **rcorr( )** no pacote **Hmisc** produz correlações e covariâncias e seus níveis de significância para as correlações de Pearson e de Sperman. Entretanto, o *input* precisa ser uma matriz e o método de exclusão *pairwise* é utilizado.

```
# Correlations/covariances among numeric variables in
# data frame mtcars. Use listwise deletion of missing data.
cor(mtcars, use="complete.obs", method="kendall")
cov(mtcars, use="complete.obs")

# Correlations with significance levels
require(Hmisc)
rcorr(x,y, type="pearson") # type can be pearson or spearman

#mtcars is a data frame
rcorr(as.matrix(mtcars))
```

Utilize a função **corrgram( )** para gerar correlogramas e **pairs( )** ou **splom( )** para criar matrizes de diagramas de dispersão.

Testes de comparação de grupos paramétricos e não-paramétricos

Testes paramétricos

A função **t.test( )* produz uma série de testes t. Diferentemente da maior parte dos pacotes estatísticos, o padrão assume variâncias diferentes a aplica a correção dos graus de liberdade de Welsh.

```
# independent 2-group t-test
t.test(y~x) # where y is numeric and x is a binary factor

# independent 2-group t-test
t.test(y1,y2) # where y1 and y2 are numeric

# paired t-test
t.test(y1,y2,paired=TRUE) # where y1 & y2 are numeric

# one sample t-test
t.test(y,mu=3) # Ho: mu=3

# anova one-way
aov(y~x) # where y is numeric and x is a factor

# anova one-way
foo <- lm(y~x) # where y is numeric and x is a factor
anova(foo)
```

###Testes não paramétricos

R oferece funções para executar os testes Mann-Whitney U, Wilcoxon Signed Rank, Kruskal Wallis e Friedman.

```
# independent 2-group Mann-Whitney U Test
wilcox.test(y~A)
# where y is numeric and A is A binary factor

# independent 2-group Mann-Whitney U Test
wilcox.test(y,x) # where y and x are numeric

# dependent 2-group Wilcoxon Signed Rank Test
wilcox.test(y1,y2,paired=TRUE) # where y1 and y2 are numeric

# Kruskal Wallis Test One Way Anova by Ranks
kruskal.test(y~A) # where y1 is numeric and A is a factor

# Randomized Block Design - Friedman Test
friedman.test(y~A|B)
# where y are the data values, A is a grouping factor and B is a blocking factor
```

For the wilcox.test you can use the alternative="less" or alternative="greater" option to specify a one tailed test.

Visualizando os resultados

Além do histograma, que já utilizamos para observar a normalidade, o R proporciona um conjunto de outros tipo de gráficos: Barplot, Dispersão, Box plots, Pie Chart. Os exemplos abaixo utilizam os dados carregados anteriormente nesta aula. Cole os comandos no console do R para verificar os resultados.

```
# Simple Bar Plot
counts <- table(mtcars$gear)
barplot(counts, main="Car Distribution", xlab="Number of Gears")

# Simple Horizontal Bar Plot with Added Labels
counts <- table(mtcars$gear)
barplot(counts, main="Car Distribution", horiz=TRUE,
  names.arg=c("3 Gears", "4 Gears", "5 Gears"))

# Grouped Bar Plot
counts <- table(mtcars$vs, mtcars$gear)
barplot(counts, main="Car Distribution by Gears and VS",
  xlab="Number of Gears", col=c("darkblue","red"),
  legend = rownames(counts), beside=TRUE)

# Simple Scatterplot
attach(mtcars)
plot(wt, mpg, main="Scatterplot Example",
   xlab="Car Weight ", ylab="Miles Per Gallon ", pch=19)
# Add fit lines
abline(lm(mpg~wt), col="red") # regression line (y~x)
lines(lowess(wt,mpg), col="blue") # lowess line (x,y)
```

```r
# Basic Scatterplot Matrix
pairs(~mpg+disp+drat+wt,data=mtcars,
   main="Simple Scatterplot Matrix")

# Boxplot of MPG by Car Cylinders
boxplot(mpg~cyl,data=mtcars, main="Car Milage Data",
   xlab="Number of Cylinders", ylab="Miles Per Gallon")

# Simple Pie Chart
slices <- c(10, 12,4, 16, 8)
lbls <- c("US", "UK", "Australia", "Germany", "France")
pie(slices, labels = lbls, main="Pie Chart of Countries")

# Pie Chart with Percentages
slices <- c(10, 12, 4, 16, 8)
lbls <- c("US", "UK", "Australia", "Germany", "France")
pct <- round(slices/sum(slices)*100)
lbls <- paste(lbls, pct) # add percents to labels
lbls <- paste(lbls,"%",sep="") # ad % to labels
pie(slices,labels = lbls, col=rainbow(length(lbls)),
   main="Pie Chart of Countries")
```

Atividades

1. Considere a base de dados **mtcars** utilizada nesta atividade. Descreva as variáveis categóricas **cyl**, **vs** e **gear** por meio de gráficos de colunas (barplot).
2. Verifique se existe associação significativa entre **cyl** e **gear**. Proceda um teste de hipóteses e apresente um gráfico de colunas das duas variáveis em conjunto.
3. Verifique se o consumo (**mpg**) adere a um modelo de distribuição normal.
4. Considerando os resultados obtidos na questão 3, proceda um teste um teste de hipóteses para verificar se **vs** impacta significativamente em **mpg**. Repita para **gear**.