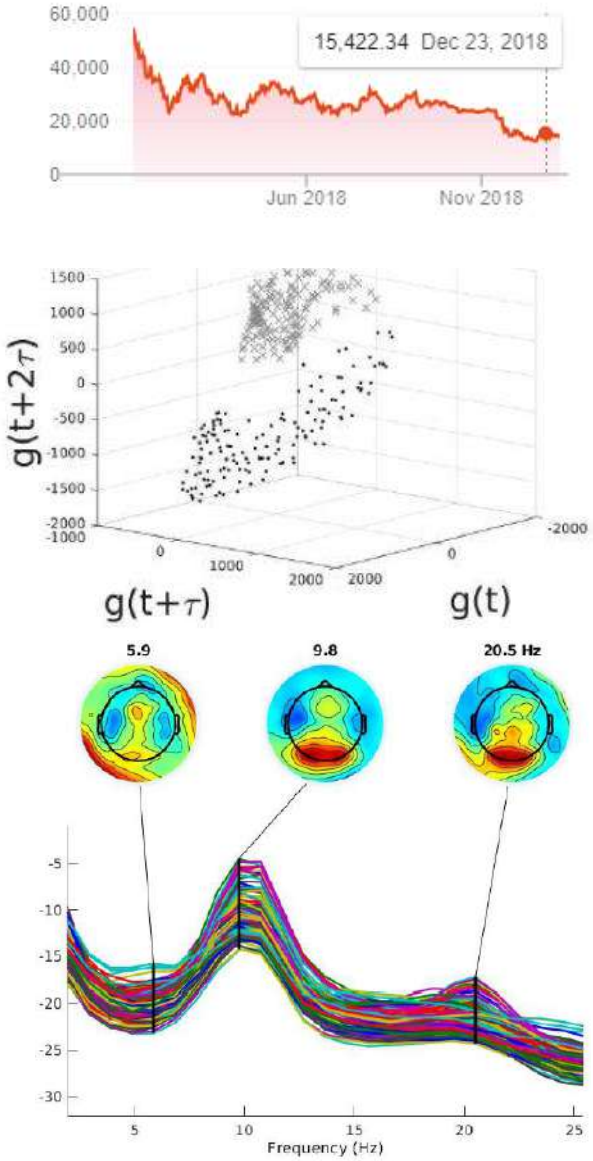


UNICAMP

Instituto de Computação

FAPESP
FUNDAÇÃO DE AMPARO À PESQUISA DO ESTADO DE SÃO PAULO

NeuroMat

Introdução à análise de séries temporais: 3. Análise multivariada

ARTHUR VALENCIO

Pós-doutorando IC/Unicamp

CEPID NeuroMat

Projeto TOPE Unicamp, Campinas, 14 de Outubro de 2019

>>Resumo do curso

Dia 1: Introdução à ferramentas de análise de dados

- Planilhas
- Linguagens de programação de propósito geral
- Linguagens de programação voltadas à dados

Dia 2: Técnicas convencionais de análise

- análise espectral
- wavelet
- correlação de Pearson e Spearman
- regressão

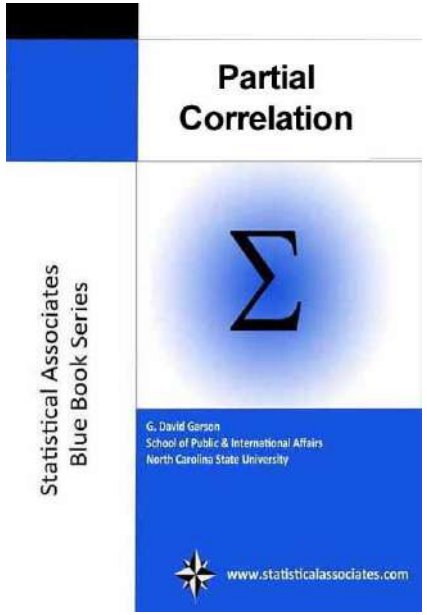
Dia 3: Análise multivariada

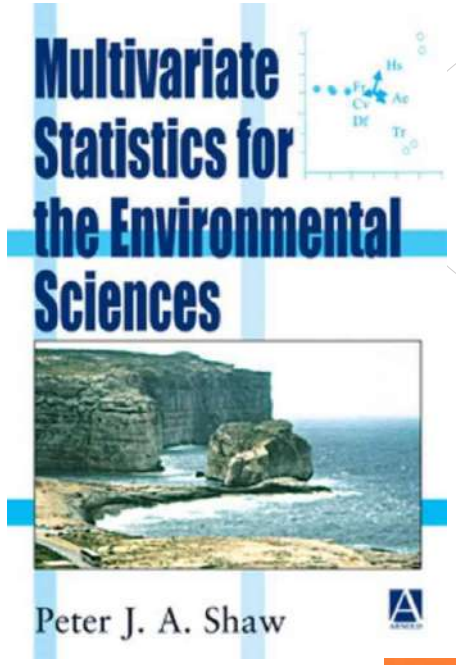
- correlação parcial
- análise fatorial
- análise de componentes principais
- análise de componentes independentes

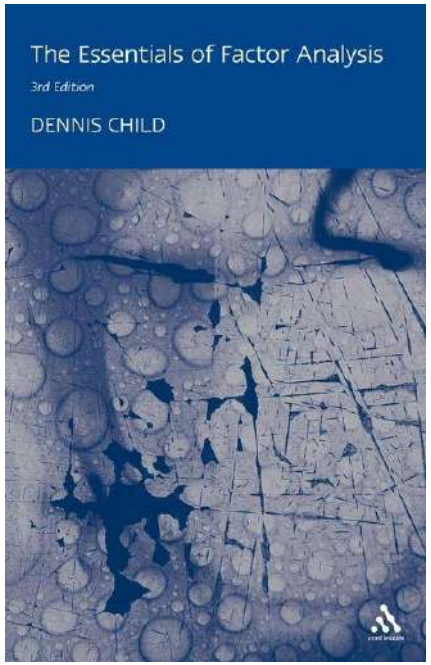
Dia 4: Séries não-lineares e teoria da informação

- reconstrução do espaço de fase
- expoentes de Lyapunov
- entropia de Shannon
- informação mútua, entropia de transferência, CaMI, direcionalidade

>>Quick refs


Partial
Correlation
Statistical Associates
Blue Book Series
G. David Garson
School of Public & International Affairs
North Carolina State University
www.statisticalassociates.com


Multivariate
Statistics for
the Environmental
Sciences
Peter J. A. Shaw


The Essentials of Factor Analysis
3rd Edition
DENNIS CHILD


INDEPENDENT COMPONENT ANALYSIS
A Tutorial Introduction
James V. Stone

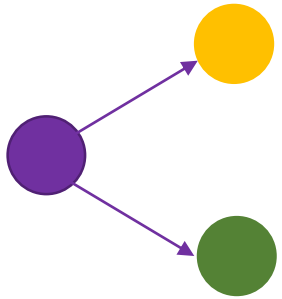
>>Objetivo

Dado que fizemos uma pesquisa coletando todas as *n* variáveis que podíamos pensar, relacionadas ao Sistema, queremos:

Descobrir quais variáveis estão relacionadas e quais não

>>Correlação parcial

Idéia: Qual é o grau de associação entre duas variáveis, se removermos o efeito produzido por todas as outras variáveis que também as influenciam?



Queremos:

- $|r|$ alto entre roxo e os demais
- $|r|$ baixo entre amarelo e verde

Princípio:

1. Fazer regressão linear entre as variáveis de análise (de interesse) e as variáveis de controle (que queremos remover)
2. Ver quais são os resíduos (i.e. os “erros” do ajuste dos dados)
3. Calcular a correlação dos resíduos

>>Correlação parcial

Matlab:

r_xy=partialcorr(x,y,z)

-- (z é a(s) variável(s) de controle)

-- exige Statistics and Machine Learning Toolbox

Todas as outras linguagens:

Vc tem que implementar a partir do princípio

Não é difícil. P. ex.:

R:

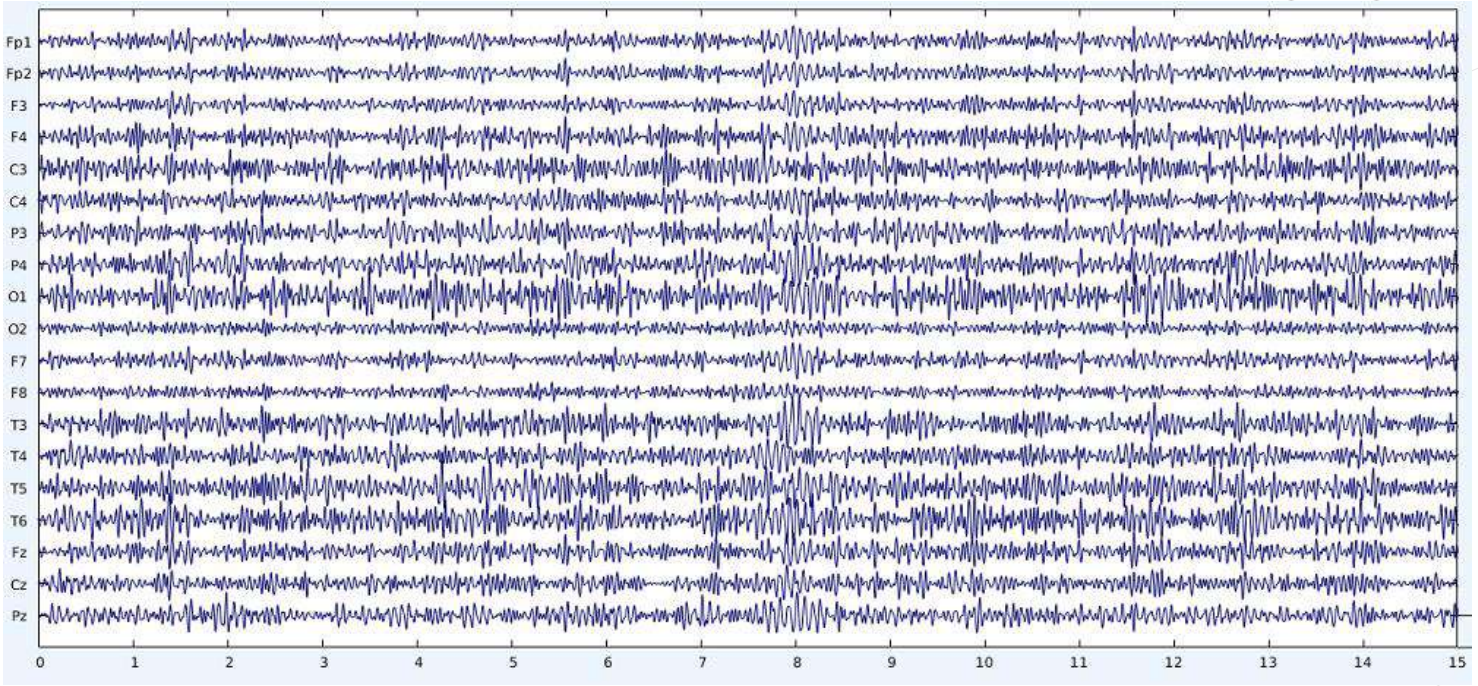
```
modelo1=lm(x~z)
residuo1=modelo1$residuals
modelo2=lm(y~z)
residuo2=modelo2$residuals
r_xy=cor(residuo1,residuo2)
```

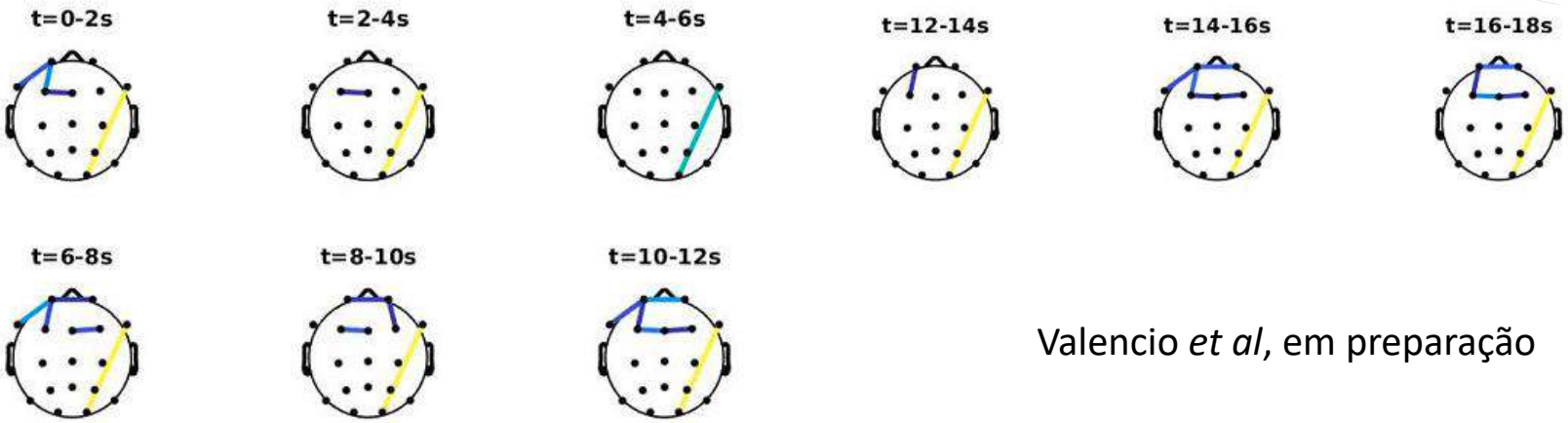
→ Faça a regressão linear

→ Calcule os resíduos

→ Faça a correlação dos resíduos

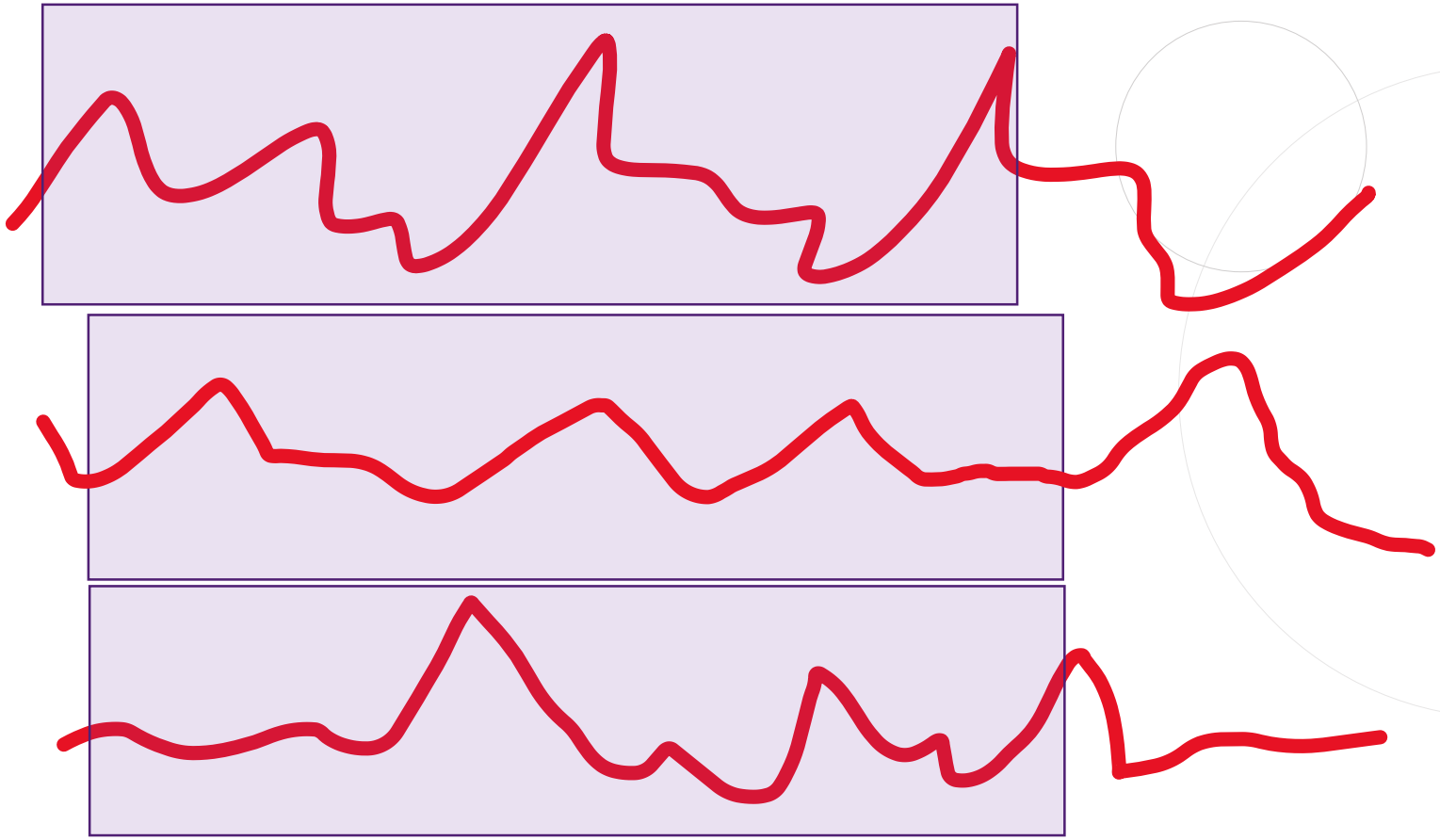
>>Correlação parcial: aplicação

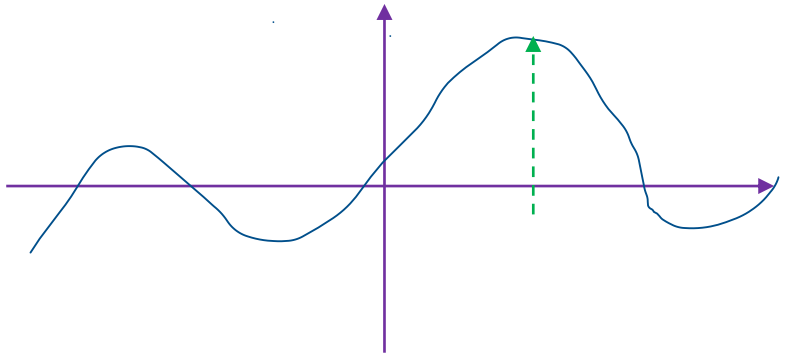




Valencio *et al*, em preparação

>>Correlação parcial cruzada





>>Análise fatorial (exploratória)

Spearman (1904)
(buscando identificar padrões nas notas dos alunos de colégio)

O nosso problema é mais complicado do que identificar/remover co-fatores:

- Temos um Sistema no qual algumas (*n*) variáveis agem conjuntamente formando um grupo, outras (*k*) formam outro, e assim por diante

- Mas não sabemos:

  (a) quantos grupos são

  (b) que variáveis compõem estes grupos

Para simplificar: Sistema sem outliers, amostra maior do que número de fatores, não há colinearidade perfeita, etc etc etc

>>Análise fatorial

Cada **variável** é uma combinação linear de um número limitado (m) de **Fatores** latentes ("escondidos") mais termos de erro:

$$X_i = a_{i,0} + a_{i,1}F_1 + a_{i,2}F_2 + \cdots + a_{i,m}F_m + \varepsilon$$

- Usamos a máxima verossimilhança dos dados com o modelo para identificar a composição dos fatores
- Usamos a variância para identificar o número de fatores necessários

Na prática:

Python:

```
import pandas as pd
import matplotlib.pyplot as plt
from factor_analyzer import FactorAnalyzer
```

Importar bibliotecas

```
fa=FactorAnalyzer()
fa.analyze=(dados,14,rotation=None)
autovalores, v = fa.get_eigenvalues()
```

Testar alguns modelos, crescendo número de fatores até 14

```
plt.plot(range(1,df.shape[1]+1),autovalores)
plt.xlabel('Número de fatores')
plt.ylabel('Autovalores')
plt.show()
```

Fazer gráfico

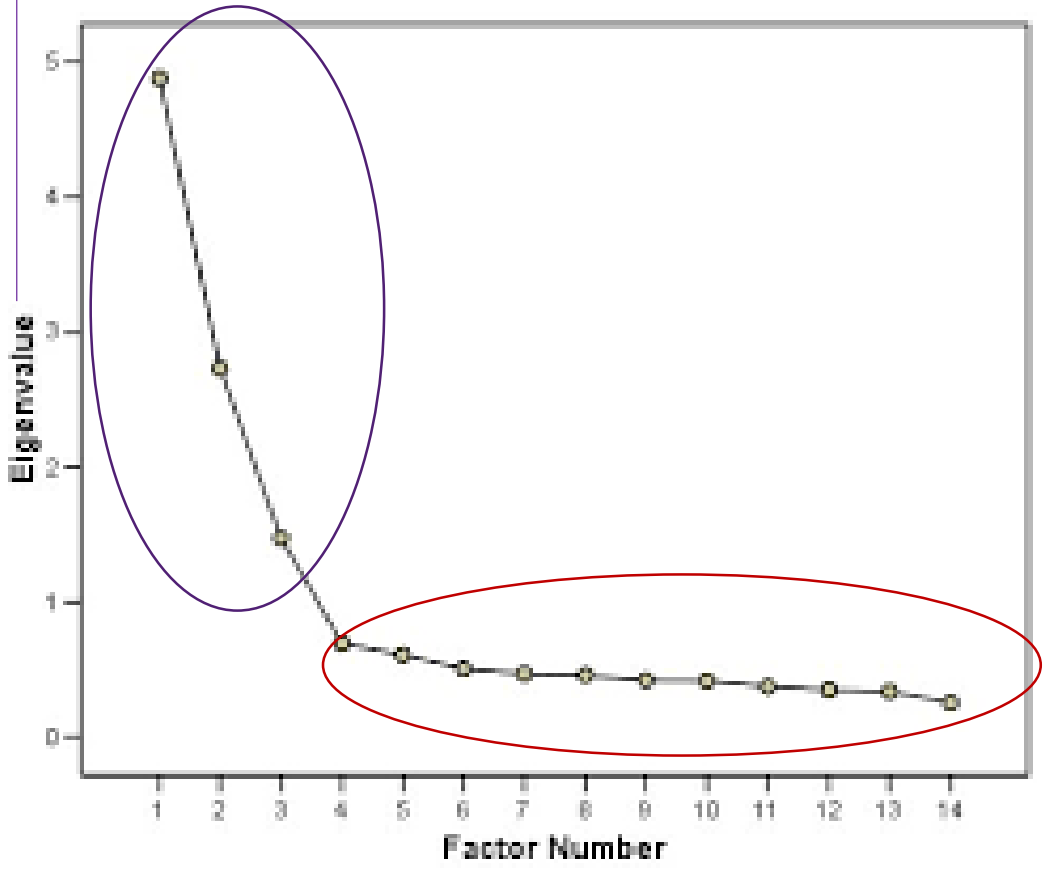
>>Análise fatorial

Cattell (1966) – Scree plot

Obs: existem outros métodos de seleção:

- Regra de Kaiser: linha de corte no autovalor = 1
- Medidas como VSS, MAP, etc

Autovalor A_i= variância total explicada pelo fator *i*



$$\sum_i A_i = N_{fatores}$$

- 5/14=**35,7%** da variância é explicada pelo fator 1
- 2,8/14=**20%** explicada pelo fator 2
- 1,5/14=**10,7%** explicada pelo fator 3

>>Análise fatorial

Já que só 3 fatores importam:

Continuando Python:
fa=FactorAnalyzer()
fa.analyze(df,3,rotation="varimax")
fa.loadings

Refazer a conta para 3 fatores, agora escolher direção com maior variancia

Mostrar a tabela de verossimilhança que permite identificar que variáveis compõem os fatores

>>Análise fatorial: aplicação

EFA Maximum Likelihood Estimate

	Factor 1	Factor 2	Factor 3
Number of disaster decrees	*-0.3568*	-0.1408	-0.0015
GDP per capita	0.2546	**0.9643**	-0.0170
Gini	**-0.5135**	-0.0240	0.2782
HDI	**0.9051**	0.2567	0.2000
Tax revenue per capita	0.1772	**0.6722**	0.1788
Population group	-0.0742	0.1339	**0.8771**
Federation unit	**0.6430**	0.1663	-0.1229

Valencio, Valencio e Baptista, *submetido*

>>Análise fatorial: aplicação

Ex: Torneio de tênis

	Fator 1	Fator 2	Fator 3
Número de vitórias	**0.9**	0.3	0.1
Velocidade do saque	**0.75**	0.3	0.1
Velocidade na quadra	**0.75**	**0.85**	0.2
Altura	0.3	**0.6**	**0.9**
Peso	0.1	-0.25	**0.85**
Idade	0.2	**-0.6**	-0.4

Fator “desempenho”

Fator “agilidade”

Fator “estrutura física”

- Vitória associada à velocidade do saque e na quadra
- Velocidade na quadra possivelmente associada à idade e altura
- Altura associada ao peso

(resultado meramente ilustrativo)

>>Análise fatorial confirmatória

- Ok, já descobri meus grupos de variáveis relacionadas

(seja por seu resultado de análise factorial exploratória, seja por teoria da sua área)

➢ Agora:

❖ Como confirmar com novos conjuntos de dados a minha teoria?

❖ Utilizando o princípio do janelamento, este procedimento pode ser utilizado para ver o quanto a teoria se adequa ao longo do tempo

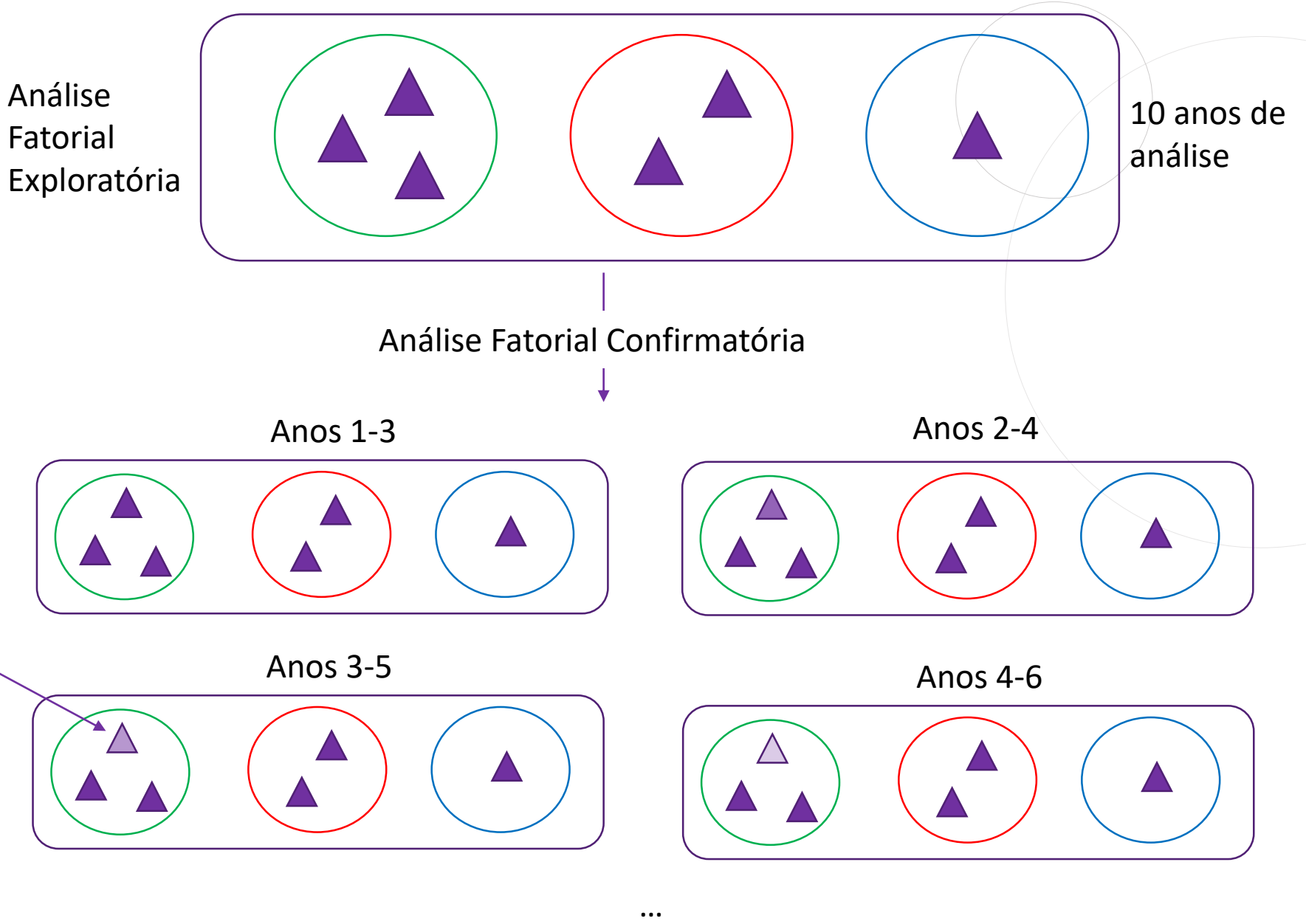
Python:

```
import pandas as pd
from factor_analyzer import (ConfirmatoryFactorAnalyzer, ModelSpecificationParser)
model_factors={“F1”:[“V1”, “V2”,”V3”],
                        “F2”:[ “V3”,”V4”,”V6”],
                         “F3”:[“V4”,”V5”]}
model_spec = ModelSpecificationParser.parse_model_specification_from_dict(data,model_factors)
cfa=ConfirmatoryFactorAnalyzer(model_spec,disp=False)
cfa.fit(data.values)
cfa.loadings_
```

→

- Ele vai te mostrar aquela tabela de máxima verossimilhança.
- Mas: ele vai forçar que elementos fora do modelo tenham valor zero. Assim, você pode como seu modelo se adequa aos dados

>>Análise fatorial confirmatória: aplicação



A contribuição desta variável vai diminuindo (progressiva ou repentinamente)

p. ex.: nova política de tratamento de esgoto faz com que tamanho da população não seja mais fator significativo na qualidade de água

>>Análise de componentes principais (PCA)

Pearson (1901) & Hotelling (c. 1930)

Idéia: Transformar os nossos dados **X** em novas variáveis **Y** não correlacionadas

Componentes principais

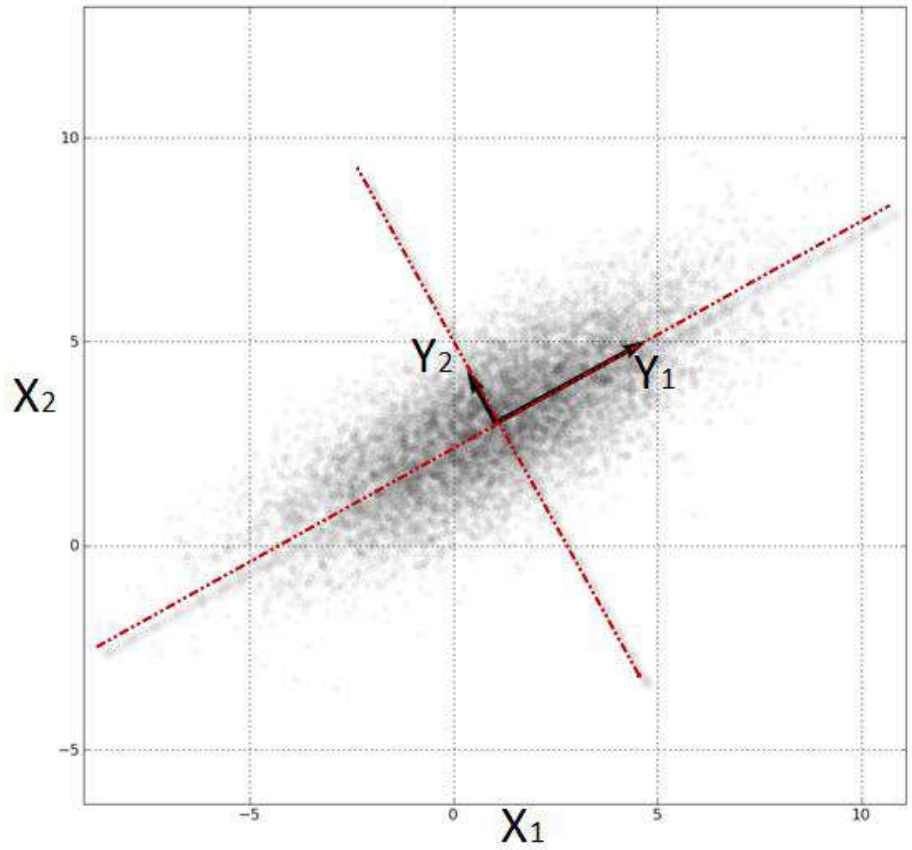
- O primeiro componente principal será aquele que modela a maior variabilidade dos dados
- Os demais estarão em ordem decrescente de variância **com a condição de serem ortogonais aos demais**

Em Álgebra Linear, $\mathbf{Y} = \mathbf{XW}$,
aonde **W** é a matriz de pesos, com colunas definidas por $\mathbf{X^T X}$

- Na análise factorial os dados eram modelados como uma combinação dos fatores
- Aqui são os componentes que são modelados como uma combinação dos dados

>>Análise de componentes principais (PCA)

De modo simplista, queremos achar o novo eixo que faz com que os dados não estejam mais correlacionados:



E, então, vemos o quanto de cada variável original compõe os novos eixos

>>Análise de componentes principais (PCA)

Matlab (Statistics toolbox):
coef=pca(dados)

Octave (Statistics package):
coef=princomp(dados)

R:
dados.pca<-prcomp(dados,center=TRUE,scale.=TRUE)

Excel: instalar NumXL add-in

Python:

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
dados=StandardScaler().fit_transform(dados)
pca=PCA(n_components=2)
componentes=pca.fit_transform(dados)
componentes_dataframe=pd.DataFrame(data=componentes, columns=['Comp. 1', 'Comp. 2'])
componentes_dataframe()
```

bibliotecas

Normalização dos dados

PCA

Fazer tabela

>>Análise de componentes independentes

Hérault, Ans, Jutten, Comon, Bell, Sejnowski, Hyvärinen, Oja (1984-1995)

Idéia: Novamente queremos passer os dados **X** para um Sistema **Y**, mas não queremos só que o novo sistema **Y** seja formado somente por variáveis não correlacionadas → queremos que seja de variáveis **independentes**

Princípio: as variáveis são compostas de fontes que queremos identificar. Estas fontes são: independentes e com distribuição não-gaussiana.

Portanto:

- vamos minimizar a informação mútua (próx aula!) entre as variáveis do novo sistema
- Vamos maximizar a não-gaussianidade (medida de curtose etc) dessas variáveis

>>Análise de componentes independents (ICA)

Hérault, Ans, Jutten, Comon, Bell, Sejnowski, Hyvärinen, Oja (1984-1995)

Idéia: Novamente queremos passer os dados **X** para um Sistema **Y**, mas não queremos só que o novo sistema **Y** seja formado somente por variáveis não correlacionadas → queremos que seja de variáveis **independentes**

Princípio: as variáveis são compostas de fontes que queremos identificar. Estas fontes são: independentes e com distribuição não-gaussiana.

Portanto:

- vamos minimizar a informação mútua (próx aula!) entre as variáveis do novo sistema
- Vamos maximizar a não-gaussianidade (medida de curtose etc) dessas variáveis

>>Análise de componentes independentes

Python:

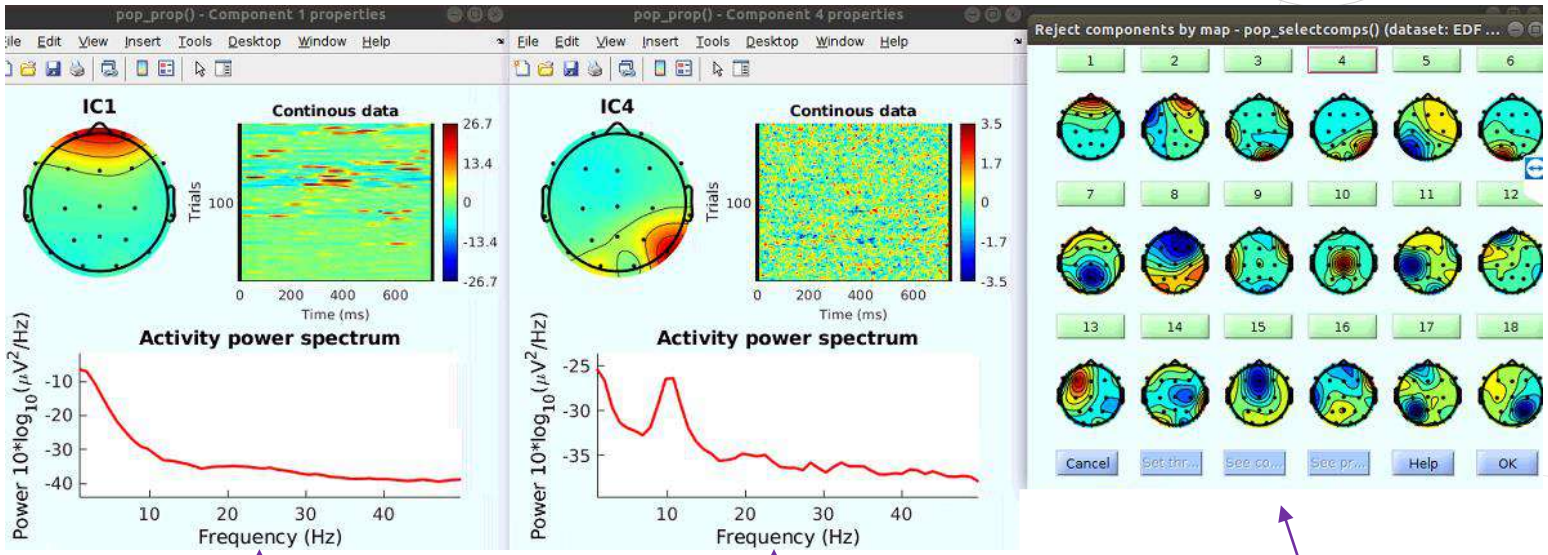
```
from sklearn.decomposition import FastICA
ica=FastICA(n_components=3)
componentes=ica.fit_transform(dados)
```

Nas demais linguagens você precisará encontrar soluções específicas para o seu caso desenvolvidas pela comunidade

Ex: Pacote EEGLAB para Matlab possui rotinas de ICA para cálculo em sinais de EEG

>>Análise de componentes independents: aplicação

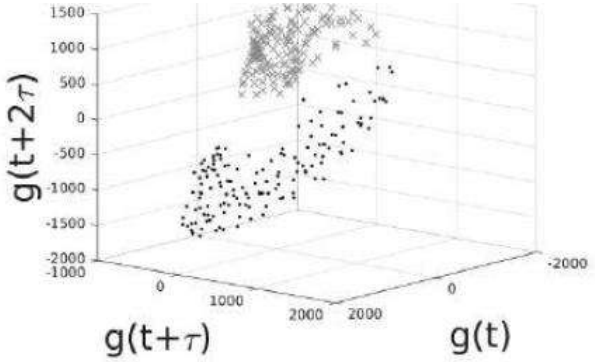
Resultados do ICA implementado no EEGLAB (para Matlab)

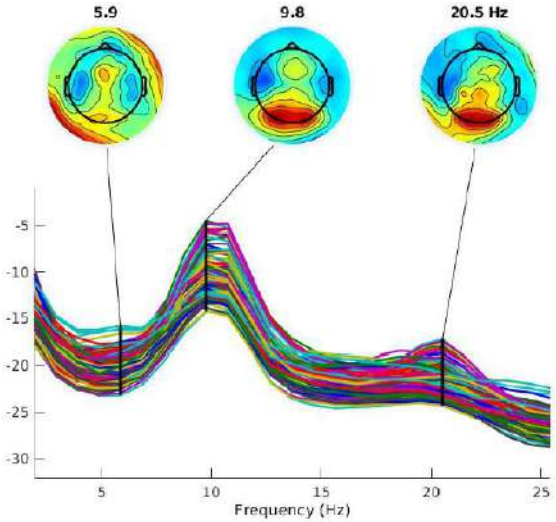


Piscadas

Atividade cerebral

Posso comparar todas as components e excluir aquelas que representam efeitos indesejados, fontes de ruído, etc





UNICAMP

Instituto de Computação

FAPESP
FUNDAÇÃO DE AMPARO À PESQUISA DO ESTADO DE SÃO PAULO

NeuroMat

Obrigado!

arthur_valencio@physics.org

http://www.arthurvalencio.com
http://neuromat.numec.prp.usp.br

A.V. agradece à FAPESP por uma bolsa de pós-doutoramento (#2018/09900-8).
CEPID NeuroMat agradece apoio FAPESP (#2013/07699-0).