Course Discussion Progress Resources

All Topics

Add a Post Search all posts Search

Filter Topics

filter topics

All Discussions

Posts I'm Following

(Optional) Unit 8 Principal component analysis:(Optional) Lecture 23: Principal Component Analysis

1. Objectives

2. Introduction

3. Multivariate Statistics and Geometry Behind the Empirical Covariance

4. Geometry Behind the Empirical Covariance Matrix - Continued

5. Review of Linear Algebra Required for this Lecture

Resolution of the Final Exam in Julia

discussion posted 2 days ago by Lobianco

The resolution of Final Exam (except theoretical-only questions) in Julia (copyable code at the bottom) :

6. Principal Component Analysis (PCA) - Theorem

7. Largest Eigenvalue and Principal Directions

```
# Final Exam of MITx 18.6051x, Dicember 2019

using Distributions, SymPy, LinearAlgebra, CSV, DataFrames,TableView   ✓

# ******************************************************************************
# ** Page 2 - Review of fundamentals *******************************************
# ******************************************************************************

# ******************************************************************************
# * Square of a standard normal: Warmup
# Just the Chi Square with 1 degree of freedom

# ******************************************************************************
# * Approximation via Central Limit Theorem

χ1 = Chisq(1) # T-distribution   Chisq{Float64}(ν=1.0)
α = 0.05   0.0500
q = quantile(χ1,1-α)   3.84…

# Method #1: using CLT:
A100 = Normal(1,sqrt(2/100))   Normal{Float64}(μ=1.0, σ=0.1414213562373095)
A16 = Normal(1,sqrt(2/10^6))   Normal{Float64}(μ=1.0, σ=0.001414213562373095)
q = quantile(A100,1-α)   1.23…
q = quantile(A16,1-α)   1.00…

# Method 2: true values using xhi-squared distribution without the  CLT
# approximation:
χ100 = Chisq(100)   Chisq{Float64}(ν=100.0)
q = quantile(χ100,1-α)/100   1.24…
χ16 = Chisq(10^6)   Chisq{Float64}(ν=1.0e6)
q = quantile(χ16,1-α)/10^6   1.00…
```

```
# ******************************************************************************
# ** Page 3 - Exponential family ***********************************************
# ******************************************************************************

# ******************************************************************************
# * Canonical form
θ = symbols("θ", real=true, negative=true)   θ
μ, x, Σx, x̄, n = symbols("μ x Σx x̄ n", positive= true, real=true)   (μ, x, Σx, x̄, n)
# Original function..
fμ = (1/sympy.sqrt(2*sympy.pi*x^3))*sympy.exp(-(x-μ)^2/(2*μ^2*x))   sqrt(2)*exp(-(x - μ)^2/(2*x*μ^2))/(2*sqrt(pi)*x^(3/2))
# Canonical exponential form..
θExpr = -1/(2*μ^2)   -1/(2*μ^2)
bθExpr = -sympy.sqrt(-2*θ)   -sqrt(2)*sqrt(-θ)
cθExpr = -1/(2*x)+sympy.log(1/sympy.sqrt(2*sympy.pi*x^3))   log(sqrt(2)/(2*sqrt(pi)*x^(3/2))) - 1/(2*x)
fθ = exp(x*θ-bθExpr+cθExpr)   sqrt(2)*exp(x*θ + sqrt(2)*sqrt(-θ) - 1/(2*x))/(2*sqrt(pi)*x^(3/2))

# Checks the two versions are the same
```

```
fμ2 = subs(fθ,Dict(θ => θExpr))  | sqrt(2)*exp(-x/(2*μ^2) + 1/μ - 1/(2*x))/(2*sqrt(pi)*x^(3/2))
test = simplify(fμ - fμ2)  | 0

# ******************************************************************************
# * Canonical link
gExpr = solve(Eq(θExpr,θ),μ)[1]  | sqrt(2)/(2*sqrt(-θ))

# ******************************************************************************
# * Expectation and variance
Eθ = diff(bθExpr,θ)  | -sqrt(2)*sqrt(-θ)/(2*θ)
Eμ = subs(Eθ,Dict(θ => θExpr))  | μ
Varθ = diff(Eθ,θ)  | sqrt(2)*sqrt(-θ)/(4*θ^2)
Varμ = subs(Varθ,Dict(θ => θExpr))  | μ^3

# ******************************************************************************
# * Fisher Information
l1  = sympy.log(fθ)  | log(sqrt(2)*exp(x*θ + sqrt(2)*sqrt(-θ) - 1/(2*x))/(2*sqrt(pi)*x^(3/2)))
dl1 = simplify(diff(l1,θ))  | x + sqrt(2)*sqrt(-θ)/(2*θ)
dl2 = simplify(diff(dl1,θ)).as_real_imag()[1]  | sqrt(2)/(4*(-θ)^(3/2))
FisherIθ = simplify(subs(-dl2,Dict(x=>Eθ)))  | -sqrt(2)/(4*(-θ)^(3/2))
FisherIμ = subs(FisherIθ,Dict(θ => θExpr)) # Still I of θ, even if expressed in terms of μ  | -μ^3

# ******************************************************************************
# * MLE
Lnθ = exp(θ*Σx-n*bθExpr) # omitting c(x,phi)  | exp(sqrt(2)*n*sqrt(-θ) + Σx*θ)
lnθ = log(Lnθ)  | sqrt(2)*n*sqrt(-θ) + Σx*θ
dlnθ = diff(lnθ,θ)  | sqrt(2)*n*sqrt(-θ)/(2*θ) + Σx
θhat = subs(solve(dlnθ, θ)[1], n/Σx => 1/x̄)  | -1/(2*x̄^2)

Lnμ = exp(-Σx/(2*μ^2)+n/μ) # omitting c(x,phi)  | exp(n/μ - Σx/(2*μ^2))
lnμ = log(Lnμ)  | n/μ - Σx/(2*μ^2)
dlnμ = diff(lnμ,μ)  | -n/μ^2 + Σx/μ^3
μhat = subs(solve(dlnμ, μ)[1], Σx/n => x̄)  | x̄

# ******************************************************************************
# * Asymtotic variance
# We already have the variance of θhat, but we don't have those of μhat
L1μ = exp(-x/(2*μ^2)+1/μ) # omitting c(x,phi)  | exp(-x/(2*μ^2) + 1/μ)
l1μ = log(L1μ)  | -x/(2*μ^2) + 1/μ
dl1μ = diff(l1μ,μ)  | x/μ^3 - 1/μ^2
dl2μ = diff(dl1μ,μ)  | -3*x/μ^4 + 2/μ^3
FisherIμ2 = simplify(subs(-dl2μ,Dict(x=>Eμ))) # This is the Fisher information of μ, expressed in terms of μ  | μ^(-3)
Varμ = 1/FisherIμ2  | μ^3
```

```
# ****************************************************************************
# ** Page 4 - Voting regression ***********************************************
# ****************************************************************************

# ****************************************************************************
# * Covariance matrix
# Just checking which option result in a scalar
A=rand(5,5)    > 5×5 Array{Float64,2}:
u = [0,0,0,-1,-1]    > Vector{Int64} with 5 elements
d4 = u'*A*u    2.24…
d5 = u*A*u'    > DimensionMismatch("matrix A has dimensions (5,1), matrix B has dimensions (5,5)")

# ****************************************************************************
# * Estimate the Variance
cd(@__DIR__)    ✓
df = CSV.read("data_gerber_trunc.csv",delim=",")    > 20000×16 DataFrame. Omitted printing of 8 columns
showtable(df)    > WebIO.Scope
n = size(df)[1]    20000
X = hcat(ones(n),df.civicduty,df.hawthorne,df.self,df.neighbors)    > 20000×5 Array{Float64,2}:
Y = df.voting    > 20000-element LazyArrays.ApplyArray{Int64,1,typeof(vcat),NTuple{4,CSV.Column{Int64,Int64}}}:
β = (X' * X)^-1 * X' * Y    > Vector{Float64} with 5 elements
ϵ = Y-X*β       # Residuals    > 20000-element LazyArrays.ApplyArray{Float64,1,typeof(vcat),NTuple{4,Array{Float64,1}}}:
σ² = sum([ϵᵢ^2 for ϵᵢ in ϵ])/(n-1)    0.216…
Σ = σ² * (X' * X) ^-1    > 5×5 Array{Float64,2}:
u = [0,0,0,-1,-1]    > Vector{Int64} with 5 elements
Varβ4minusβ3 = u' * Σ * u    0.000272…
```

```
# ******************************************************************************
# ** Page 5 - Independence tests for Bernoulli / regression ********************
# ******************************************************************************

r, p, q,= symbols("r p q", positive=true, real=true)   (r, p, q)

# The key in this exercise was to recognise that XY ~ Ber(r), that it, XY = 1 if and only if X and Y are both 1

# ******************************************************************************
# * Asymptotic Variance Under the Null

Σ = [ p*(1-p) r-p*q r*(1-p) ; r-p*q q*(1-q)  r*(1-q) ; r*(1-p)  r*(1-q) r*(1-r)]   › 3×3 Array{Sym,2}:
ω = [-q,-p,1]   › Vector{Sym} with 3 elements

# note that even under the null the estimators haven't zero correlation, i.e using
# Σ₀ = [ p*(1-p) 0 0 ; 0 q*(1-q)  0 ; 0 0 p*q*(1-p*q)] would be wrong
V₀ = simplify(ω' * Σ * ω)   -4*p^2*q^2 + p^2*q + p*q^2 + 6*p*q*r - 2*p*r - 2*q*r - r^2 + r
#V₀ = simplify(subs(V₀ , Dict(r => p*q)))   p*q*(-3*p*q + p + q + 1)
V₀ = simplify(subs(V₀ , Dict(r => p*q)))   p*q*(p*q - p - q + 1)

# ******************************************************************************
# * Happiness and Being in a Relationship
n = 205+179+301+315   1000
hatp = (205+179)/(205+179+301+315)   0.384
hatq = (205+301)/(205+179+301+315)   0.506
hatr = (205)/(205+179+301+315)   0.205

v0num = N(V₀.evalf(subs=Dict(p => hatp, q => hatq, r => hatr)))   0.0591…
Tn = abs(sqrt(n)*(hatr-hatq*hatp)/sqrt(v0num))   1.39…
α = 0.05   0.0500
q = quantile(Normal(),1-(α/2))   1.96…
Tn > q   false
p_value =  2 * cdf(Normal(),-Tn)   0.1642258510754747
```

Copyable code:

```
# Final Exam of MITx 18.6051x, Dicember 2019

using Distributions, SymPy, LinearAlgebra, CSV, DataFrames,TableView

# ******************************************************************************
# ** Page 2 - Review of fundamentals ******************************************
# ******************************************************************************

# ******************************************************************************
# * Square of a standard normal: Warmup
# Just the Chi Square with 1 degree of freedom

# ******************************************************************************
# * Approximation via Central Limit Theorem

χ₁ = Chisq(1) # T-distribution
α = 0.05
q = quantile(χ₁,1-α)

# Method #1: using CLT:
A100 = Normal(1,sqrt(2/100))
A16 = Normal(1,sqrt(2/10^6))
q = quantile(A100,1-α)
q = quantile(A16,1-α)

# Method 2: true values using xhi-squared distribution without the  CLT
# approximation:
χ100 = Chisq(100)
q = quantile(χ100,1-α)/100
χ16 = Chisq(10^6)
q = quantile(χ16,1-α)/10^6

# ******************************************************************************
# ** Page 3 - Exponential family **********************************************
# ******************************************************************************

# ******************************************************************************
# * Canonical form
θ = symbols("θ", real=true, negative=true)
μ, x, Σx, x̄, n = symbols("μ x Σx x̄ n", positive= true, real=true)
# Original function..
fμ = (1/sympy.sqrt(2*sympy.pi*x^3))*sympy.exp(-(x-μ)^2/(2*μ^2*x))
# Canonical exponential form..
θExpr = -1/(2*μ^2)
```

```
bθExpr = -sympy.sqrt(-2*θ)
cθExpr = -1/(2*x)+sympy.log(1/sympy.sqrt(2*sympy.pi*x^3))
fθ = exp(x*θ-bθExpr+cθExpr)

# Checks the two versions are the same..
fμ2 = subs(fθ,Dict(θ => θExpr))
test = simplify(fμ - fμ2)

# ******************************************************************************
# * Canonical link
gExpr = solve(Eq(θExpr,θ),μ)[1]

# ******************************************************************************
# * Expectation and variance
Eθ = diff(bθExpr,θ)
Eμ = subs(Eθ,Dict(θ => θExpr))
Varθ = diff(Eθ,θ)
Varμ = subs(Varθ,Dict(θ => θExpr))

# ******************************************************************************
# * Fisher Information
l1  = sympy.log(fθ)
dl1 = simplify(diff(l1,θ))
dl2 = simplify(diff(dl1,θ)).as_real_imag()[1]
FisherIθ = simplify(subs(-dl2,Dict(x=>Eθ)))
FisherIμ = subs(FisherIθ,Dict(θ => θExpr)) # Still I of θ, even if expressed in terms of μ

# ******************************************************************************
# * MLE
Lnθ = exp(θ*Σx-n*bθExpr) # omitting c(x,phi)
lnθ = log(Lnθ)
dlnθ = diff(lnθ,θ)
θhat = subs(solve(dlnθ, θ)[1], n/Σx => 1/x̄)

Lnμ = exp(-Σx/(2*μ^2)+n/μ) # omitting c(x,phi)
lnμ = log(Lnμ)
dlnμ = diff(lnμ,μ)
μhat = subs(solve(dlnμ, μ)[1], Σx/n => x̄)

# ******************************************************************************
# * Asymtotic variance
# We already have the variance of θhat, but we don't have those of μhat
L1μ = exp(-x/(2*μ^2)+1/μ) # omitting c(x,phi)
l1μ = log(L1μ)
```

```
dl1μ = diff(l1μ,μ)
dl2μ = diff(dl1μ,μ)
FisherIμ2 = simplify(subs(-dl2μ,Dict(x=>Eμ))) # This is the Fisher information of μ, expressed in terms of μ
Varμ = 1/FisherIμ2

# ******************************************************************************
# ** Page 4 - Voting regression ***********************************************
# ******************************************************************************

# ******************************************************************************
# * Covariance matrix
# Just checking which option result in a scalar
A=rand(5,5)
u = [0,0,0,-1,-1]
d4 = u'*A*u
d5 = u*A*u'

# ******************************************************************************
# * Estimate the Variance
cd(@__DIR__)
df = CSV.read("data_gerber_trunc.csv",delim=",")
showtable(df)
n = size(df)[1]
X = hcat(ones(n),df.civicduty,df.hawthorne,df.self,df.neighbors)
Y = df.voting
β = (X' * X)^-1 * X' * Y
ϵ = Y-X*β       # Residuals
σ² = sum([ϵᵢ^2 for ϵᵢ in ϵ])/(n-1)
Σ = σ² * (X' * X) ^-1
u = [0,0,0,-1,-1]
Varβ4minusβ3 = u' * Σ * u

# ******************************************************************************
# ** Page 5 - Independence tests for Bernoulli / regression *******************
# ******************************************************************************

r, p, q,= symbols("r p q", positive=true, real=true)

# The key in this exercise was to recognise that XY ~ Ber(r), that is, XY = 1 if and only if X and Y are
both 1

# ******************************************************************************
# * Asymptotic Variance Under the Null
```

```
Σ = [ p*(1-p) r-p*q r*(1-p) ; r-p*q q*(1-q)  r*(1-q) ; r*(1-p)  r*(1-q) r*(1-r)]
ω = [-q,-p,1]

# note that even under the null the estimators haven't zero correlation, i.e using
# Σₒ = [ p*(1-p) 0 0 ; 0 q*(1-q)  0 ; 0 0 p*q*(1-p*q)] would be wrong
Vₒ = simplify(ω' * Σ * ω)
#Vₒ = simplify(subs(Vₒ , Dict(r => p*q)))
Vₒ = simplify(subs(Vₒ , Dict(r => p*q)))

# ******************************************************************************
# * Happiness and Being in a Relationship
n = 205+179+301+315
hatp = (205+179)/(205+179+301+315)
hatq = (205+301)/(205+179+301+315)
hatr = (205)/(205+179+301+315)

v0num = N(Vₒ.evalf(subs=Dict(p => hatp, q => hatq, r => hatr)))
Tn = abs(sqrt(n)*(hatr-hatq*hatp)/sqrt(v0num))
α = 0.05
q = quantile(Normal(),1-(α/2))
Tn > q
p_value =  2 * cdf(Normal(),-Tn)
```

Related to: Final exam:Final Exam / 5.
This post is visible to everyone.

0 responses

Preview

Submit