analise_provas_parciais

March 7, 2021

1 Física III-A (2020/1)

2 Análise das Provas Parciais

Inicialização de pacotes

[1]:
```
using Dates
using CSV
using DataFrames
using Statistics
using StatsBase
using Plots
```

2.0.1 Dados de entrada sobre a disciplina e o curso

- o nome da disciplina
- seu período de oferta

[2]:
```
nome_disciplina = "Física III"
periodo = "2020_1";
```

2.0.2 Dados de entrada relativos à Primeira Prova (P1)

- a abreviatura do questionário
- o caminho completo do ficheiro, em formato csv, com a tabela de dados do questionário de interesse, baixado diretamente do AVA
- o número de questões
- as eventuais correspondentes categorias (e marcadores, se for o caso)
- o limite de duração do questionário (em minutos, sem a unidade)
- o início e o final da janela de acesso ao questionário (em formato tipo "10 Feb 2021 15:00")
- o número de participantes/presentes

[3]:
```
abrev_questionario_P1 = "P1"
caminho_completo_ficheiro_original_P1 = "P1_notas.csv"
N_questoes_P1 = 20
categorias_P1 = ["Lei de Coulomb - Força elétrica - Cálculos simples"
"Lei de Coulomb - Força elétrica -> Questões conceituais"
"Campo elétrico -> Questões simples"
"Campo elétrico -> Distribuições contínuas de carga -> Argumentos de simetria"
```

```
"Campo elétrico -> Distribuições contínuas de carga -> Carga total"
"Condutores em equilíbrio eletrostático"
"Condutores em equilíbrio eletrostático -> Conceitos"
"Condutores em equilíbrio eletrostático -> Cargas em condutores"
"Questões conceituais ou de cálculos simples -> Conceitos da lei de Gauss"
"Aplicações -> Simetria planar"
"Fluxo -> Fluxo a partir da distribuição de cargas"
"Potencial de distribuições discretas -> Cálculos ou gráficos"
"Potencial de distribuições contínuas -> Questões conceituais"
"Determinação de V a partir de E -> Questões simples: conceitos, gráficos e␣
 ↪cálculos rápidos"
"Determinação de E a partir de V -> Cálculo do gradiente"
"Superfícies equipotenciais e linhas de campo"
"Condutores em equilíbrio eletrostático -> Questões conceituais"
"Capacitância -> Conceitos"
"Armazenamento de energia"
"Determinação de C -> Capacitor de placas planas e paralelas"
]
limite_duracao_P1 = 120
janela_Ini_P1 = "10 Jan 2021 15:00"
janela_Fin_P1 = "10 Jan 2021 18:00"
N_participantes_P1 = 326;
```

Processando e verificando dados de entrada: * "comprimento" de categorias deve coincidir com número de questões * janela não vazia * janela maior que limite de duração do questionário

[4]:
```
if !(@isdefined(categorias_P1) & (length(categorias_P1)==N_questoes_P1))
    error("length(categorias) não coincide com N_questoes!")
end

dtformat = DateFormat("d u y H:M")
janela_ini_P1 = DateTime(janela_Ini_P1, dtformat)
janela_fin_P1 = DateTime(janela_Fin_P1, dtformat)
if !(janela_ini_P1 < janela_fin_P1)
    error("Duração da janela de acesso vazia!")
end
if (round(janela_fin_P1 - janela_ini_P1, Minute).value < limite_duracao_P1 < 0)
    error("Limite de duração do questionário menor que janela ou negativo!")
end
```

Leitura do ficheiro e transformação em dataframe:

[5]:
```
df_P1 = DataFrame(CSV.File(caminho_completo_ficheiro_original_P1, type =␣
 ↪String))[1:end-1,:]
```

[5]:

	Vínculo	Nome	Número de identificaç
	String	String?	String?
1	- Aluno	RUY DA SILVA RAYOL NETO	119018524
2	- Aluno	THAIS MORAIS RUFFO	113039695
3	- Aluno	MATHEUS ROCHA MARQUES DE ALMEIDA	115173047
4	- Aluno	ENZO CURZIO CASSANO	118037258
5	- Aluno	EDUARDO DE JESUS CARDOSO	114141534
6	- Aluno	JOSE MANDU DE FREITAS	116197913
7	- Aluno	THAIS DE CASSIA TELES SANTOS	117133750
8	- Aluno	BERNARDO MACEIO LONTRA ARAUJO	115026583
9	- Aluno	CHARLES ALEXANDRE PEREIRA GONCALVES	114195753
10	- Aluno	JULIANA FALCAO ALVES DE CARVALHO	116074717
11	- Aluno	BERNARDO GLIOCHE MIRANDA	117085761
12	- Aluno	CAIO SILVA DE FREITAS	117032792
13	- Aluno	LETHICIA MACHADO MARQUES SANTANA	111402012
14	- Aluno	NATANAEL GOMES DE OLIVEIRA	118061956
15	- Aluno	JOAO CARLOS DOS SANTOS TORRES	117094516
16	- Aluno	LUANA DUTRA SOBRINHA DA SILVA DAMASCENO	112181936
17	- Aluno	VICTOR HUGO RODRIGUEZ LORENZO	116014343
18	- Aluno	LUCCA TEIXEIRA DA FONSECA QUINTELA COELHO MARTINS	119034033
19	- Aluno	IGOR AMORIM LEITE PEREIRA	119060050
20	- Aluno	GUILHERME BAPTISTA LUPPI SILVA	117042307
21	- Aluno	BERNARDO MARCHI	114180685
22	- Aluno	ROBERTA FERREIRA DOS SANTOS	116077375
23	- Aluno	ALICE ROCHA DOS SANTOS	114140821
24	- Aluno	IGOR FERREIRA MURRAS	118156052
25	- Aluno	VINICIUS GONCALVES VERAS	118172951
26	- Aluno	FELIPE GOMES TAPARO	119059237
27	- Aluno	RAMON MATIAS NUNES MENDONCA	119038126
28	- Aluno	PEDRO HENRIQUE GUEDES FERNANDES DE MORAES	117251817
29	- Aluno	GISELLE MARTINS FEITOSA DOS SANTOS	119047175
30	- Aluno	FELIPE HIME MIRANDA	117280531
...	...	...	...

Verificando dataframe:

[6]:
```
if N_participantes_P1 != nrow(df_P1)
    error("Número declarado de participantes não coincide com número de linhas␣
 ↪menos um do ficheiro lido!")
elseif ncol(df_P1) != (N_questoes_P1 + 9)
    error("Número de questões mais nove não coincide com o número de colunas do␣
 ↪ficheiro lido!")
end
```

Elimine as colunas desnecessárias:

[7]:
```
names(df_P1)
select!(df_P1, Not([1, 2, 4, 9]));
```

Renomeie as colunas adequadamente:

[8]:
```
new_col_names_P1 = [Symbol("Q$i") for i=1:N_questoes_P1]
new_col_names_P1 = [Symbol("Ingresso"); Symbol("Inicio"); Symbol("Termino");
    Symbol("Duracao");  Symbol("Nota"); new_col_names_P1]
names!(df_P1, new_col_names_P1);
```

Converta a vírgula decimal, ",", para o ponto decimal, "." e, ademais, o traço, "-", de ausência de nota, para "0.00":

[9]:
```
df_P1 = replace.(df_P1, ","=>".")
df_P1 = replace.(df_P1, "-"=>"0.00");
```

Identifique o ano de ingresso:

[10]:
```
transform!(df_P1, :Ingresso => ByRow(x -> "20"*x[2:3]) => :Ingresso);
```

Converta Inicio e Termino em DateTime, sem unidades (Int) e, ademais, reabasteça Duracao com o número de minutos, sem unidades (Int):

[11]:
```
df_P1 = replace.(df_P1, "janeiro"=>"Jan")
df_P1 = replace.(df_P1, "fevereiro"=>"Feb")
df_P1 = replace.(df_P1, "março"=>"Mar")
df_P1 = replace.(df_P1, "abril"=>"Apr")
df_P1 = replace.(df_P1, "maio"=>"May")
df_P1 = replace.(df_P1, "junho"=>"Jun")
df_P1 = replace.(df_P1, "julho"=>"Jul")
df_P1 = replace.(df_P1, "agosto"=>"Aug")
df_P1 = replace.(df_P1, "setembro"=>"Sep")
df_P1 = replace.(df_P1, "outubro"=>"Oct")
df_P1 = replace.(df_P1, "novembro"=>"Nov")
df_P1 = replace.(df_P1, "dezembro"=>"Dec")

dtformat = DateFormat("d u Y  H:M ")
df_P1.Inicio = DateTime.(df_P1.Inicio, dtformat)
df_P1.Termino = DateTime.(df_P1.Termino, dtformat)

transform!(df_P1, :Inicio => ByRow(x -> round(x - janela_ini_P1, Minute).value)␣
 ↪=> :Inicio)
transform!(df_P1, :Termino => ByRow(x -> round(x - janela_ini_P1, Minute).
 ↪value) => :Termino)
transform!(df_P1, [:Termino, :Inicio] => ByRow(-) => :Duracao);
```

Finalmente, converta a primeira coluna, Ingresso, em Int e todas as outras colunas remanescentes, desde Nota até Q$N_questões, em Float:

[12]:
```
for i in 5:(N_questoes_P1 + 5)
    df_P1[names(df_P1)[i]] = parse.(Float64, df_P1[names(df_P1)[i]])
end
```

```
df_P1.Ingresso = parse.(Int, df_P1.Ingresso);
```

[13]:
```
df_P1
```

[13]:

	Ingresso	Inicio	Termino	Duracao	Nota	Q1	Q2	Q3	Q4	
	Int64	Int64	Int64	Int64	Float64	Float64	Float64	Float64	Float64	
1	2019	0	120	120	7.5	0.5	0.0	0.5	0.5	...
2	2013	0	119	119	0.5	0.0	0.0	0.0	0.0	...
3	2015	0	105	105	8.0	0.5	0.0	0.5	0.5	...
4	2018	0	112	112	7.5	0.0	0.0	0.5	0.5	...
5	2014	0	74	74	8.0	0.5	0.0	0.5	0.5	...
6	2016	0	71	71	6.5	0.5	0.0	0.0	0.5	...
7	2017	0	120	120	7.0	0.5	0.0	0.0	0.5	...
8	2015	0	12	12	1.5	0.0	0.0	0.5	0.0	...
9	2014	0	83	83	2.5	0.5	0.0	0.5	0.0	...
10	2016	0	104	104	6.0	0.0	0.5	0.0	0.5	...
11	2017	0	117	117	9.0	0.5	0.5	0.5	0.5	...
12	2017	0	119	119	6.5	0.5	0.0	0.5	0.5	...
13	2011	0	102	102	6.5	0.0	0.0	0.5	0.5	...
14	2018	0	105	105	8.5	0.5	0.0	0.5	0.5	...
15	2017	0	116	116	6.5	0.5	0.5	0.5	0.5	...
16	2012	0	120	120	7.0	0.0	0.5	0.5	0.0	...
17	2016	1	78	77	7.5	0.0	0.5	0.5	0.5	...
18	2019	1	70	69	8.5	0.5	0.0	0.5	0.5	...
19	2019	1	98	97	7.0	0.5	0.0	0.5	0.5	...
20	2017	1	107	106	8.0	0.5	0.0	0.0	0.5	...
21	2014	1	101	100	7.5	0.5	0.5	0.5	0.5	...
22	2016	1	118	117	7.0	0.0	0.0	0.0	0.5	...
23	2014	1	115	114	5.0	0.0	0.5	0.0	0.0	...
24	2018	1	121	120	7.5	0.5	0.0	0.5	0.5	...
25	2018	1	80	79	4.5	0.0	0.0	0.5	0.5	...
26	2019	2	95	93	5.0	0.0	0.0	0.5	0.5	...
27	2019	2	112	110	6.5	0.5	0.5	0.5	0.5	...
28	2017	2	122	120	7.5	0.0	0.0	0.0	0.5	...
29	2019	2	99	97	7.0	0.5	0.5	0.0	0.5	...
30	2017	2	122	120	6.0	0.0	0.0	0.5	0.0	...
...	...	...	...	...	...	...	...	...	...	

2.0.3 Estatísticas da P1

[14]:
```
describe(df_P1, :mean, :std, :min, :q25, :median, :q75, :max)
```

[14]:

	variable	mean	std	min	q25	median	q75	max
	Symbol	Float64	Float64	Real	Float64	Float64	Float64	Real
1	Ingresso	2016.59	2.23876	2005	2016.0	2017.0	2018.0	2020
2	Inicio	39.7975	25.8512	0	21.0	45.0	56.0	178
3	Termino	145.911	26.623	12	127.25	150.5	167.0	180
4	Duracao	106.113	16.446	2	101.0	111.0	118.0	122
5	Nota	6.49387	1.69432	0.0	5.5	6.5	7.5	10.0
6	Q1	0.354294	0.227555	0.0	0.0	0.5	0.5	0.5
7	Q2	0.210123	0.247179	0.0	0.0	0.0	0.5	0.5
8	Q3	0.337423	0.234576	0.0	0.0	0.5	0.5	0.5
9	Q4	0.337423	0.234576	0.0	0.0	0.5	0.5	0.5
10	Q5	0.326687	0.238313	0.0	0.0	0.5	0.5	0.5
11	Q6	0.343558	0.23219	0.0	0.0	0.5	0.5	0.5
12	Q7	0.187117	0.242334	0.0	0.0	0.0	0.5	0.5
13	Q8	0.300613	0.245199	0.0	0.0	0.5	0.5	0.5
14	Q9	0.371166	0.219011	0.0	0.0	0.5	0.5	0.5
15	Q10	0.369632	0.219856	0.0	0.0	0.5	0.5	0.5
16	Q11	0.369632	0.219856	0.0	0.0	0.5	0.5	0.5
17	Q12	0.383436	0.211737	0.0	0.5	0.5	0.5	0.5
18	Q13	0.277607	0.248853	0.0	0.0	0.5	0.5	0.5
19	Q14	0.368098	0.220686	0.0	0.0	0.5	0.5	0.5
20	Q15	0.361963	0.22387	0.0	0.0	0.5	0.5	0.5
21	Q16	0.289877	0.247179	0.0	0.0	0.5	0.5	0.5
22	Q17	0.386503	0.209766	0.0	0.5	0.5	0.5	0.5
23	Q18	0.300613	0.245199	0.0	0.0	0.5	0.5	0.5
24	Q19	0.219325	0.248492	0.0	0.0	0.0	0.5	0.5
25	Q20	0.398773	0.201223	0.0	0.5	0.5	0.5	0.5

[15]:
```
modes.(eachcol(df_P1))
```

[15]:
```
25-element Array{Array{T,1} where T,1}:
 [2018]
 [0]
 [180]
 [120]
 [7.0, 7.5]
 [0.5]
 [0.0]
 [0.5]
 [0.5]
 [0.5]
 [0.5]
 [0.0]
 [0.5]
 [0.5]
 [0.5]
 [0.5]
```

```
[0.5]
[0.5]
[0.5]
[0.5]
[0.5]
[0.5]
[0.5]
[0.0]
[0.5]
```

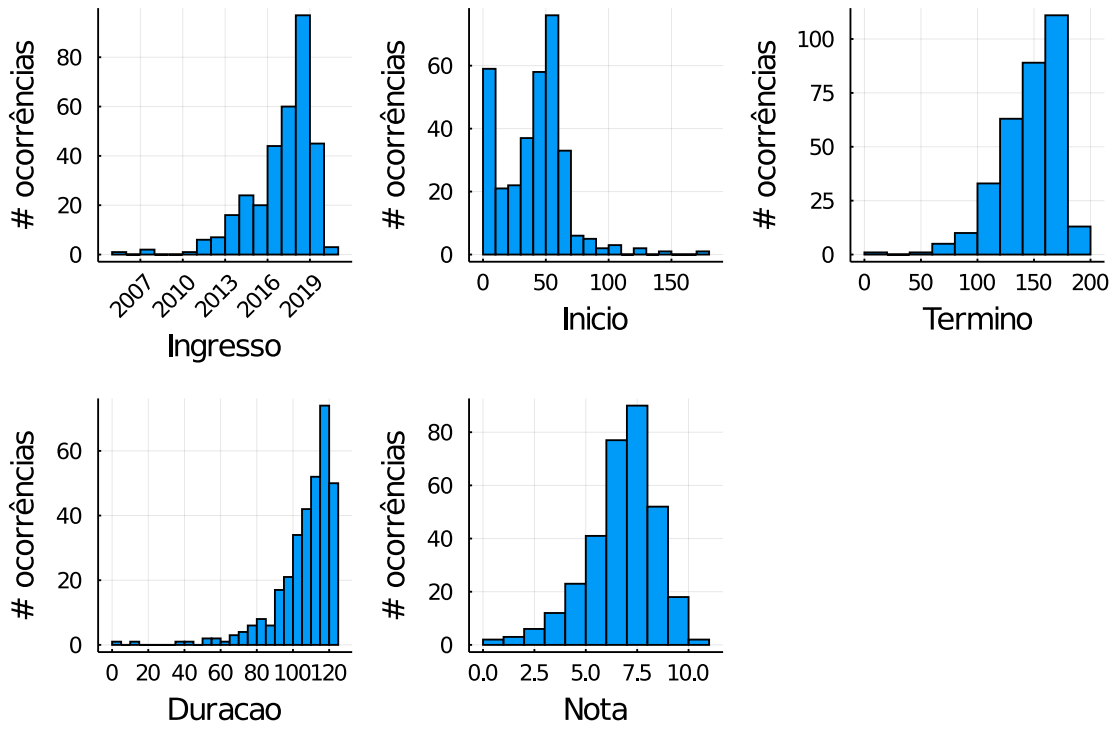
[16]:
```
#using DataVoyager
```

[17]:
```
#Voyager(df_P1)
```

[18]:
```
p_P1 = []
for i in names(df_P1)
    if i == "Ingresso"
        push!(p_P1, histogram(df_P1[i], xlabel = i, ylabel = "# ocorrências",␣
↪legend=false, xrotation = 45))
    else
        push!(p_P1, histogram(df_P1[i], xlabel = i, ylabel = "# ocorrências",␣
↪legend=false))
    end
end
```

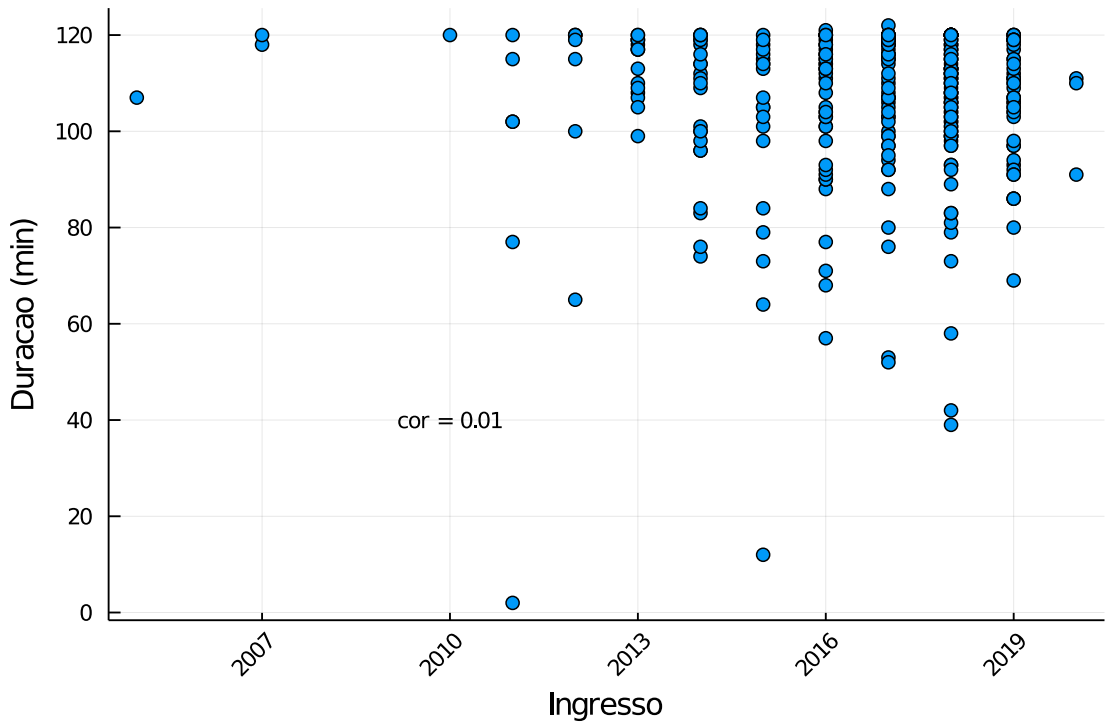
[19]:
```
plot(p_P1[1],p_P1[2],p_P1[3],p_P1[4],p_P1[5])
```

[19]:



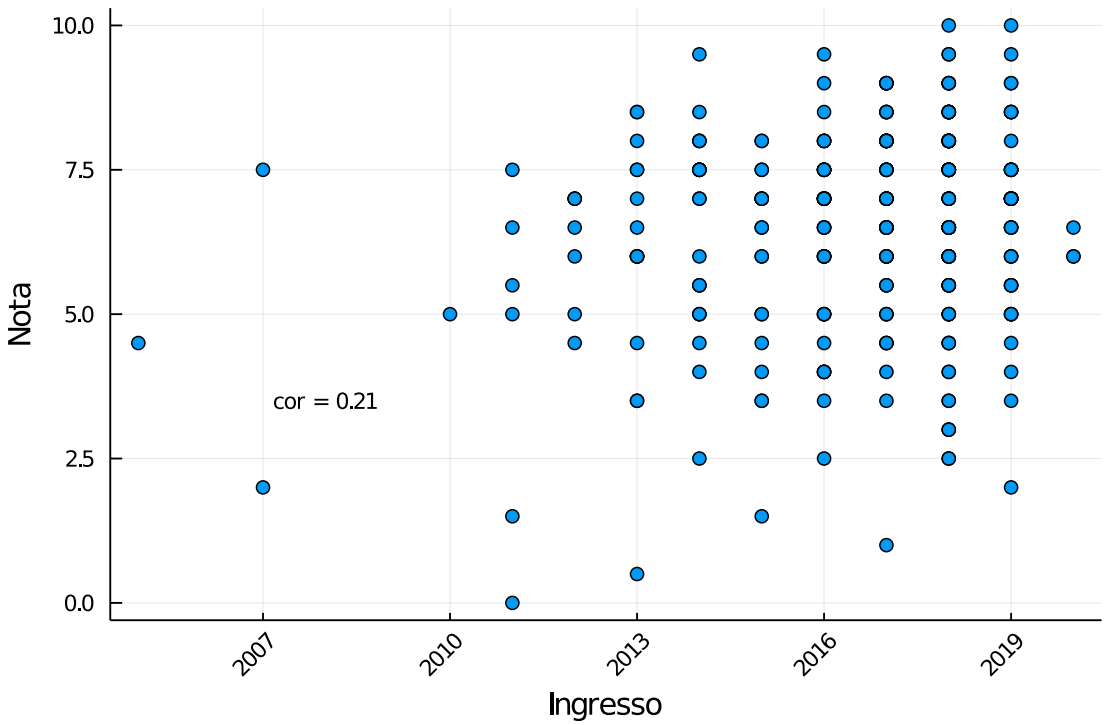
[20]:
```
scatter(df_P1.Ingresso, df_P1.Duracao, xlabel="Ingresso", ylabel="Duracao␣
 ↪(min)", xrotation=45, legend=false)
corout = cor(df_P1.Ingresso, df_P1.Duracao)
aux1=annotate!(2010, 40, text("cor = "*string(round(corout,digits=2)), 8))
```

[20]:



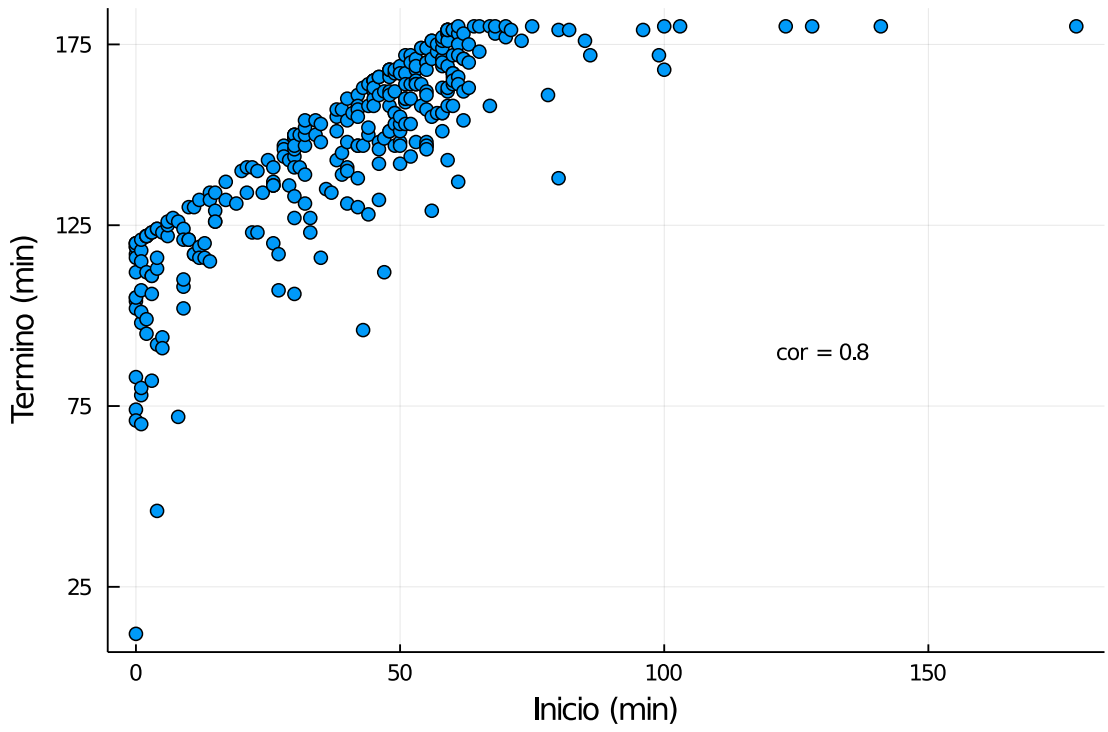
[21]:
```
scatter(df_P1.Ingresso, df_P1.Nota, xlabel="Ingresso", ylabel="Nota",␣
 ↪xrotation=45, legend=false)
corout = cor(df_P1.Ingresso, df_P1.Nota)
aux2=annotate!(2008, 3.5, text("cor = "*string(round(corout,digits=2)), 8))
```

[21]:



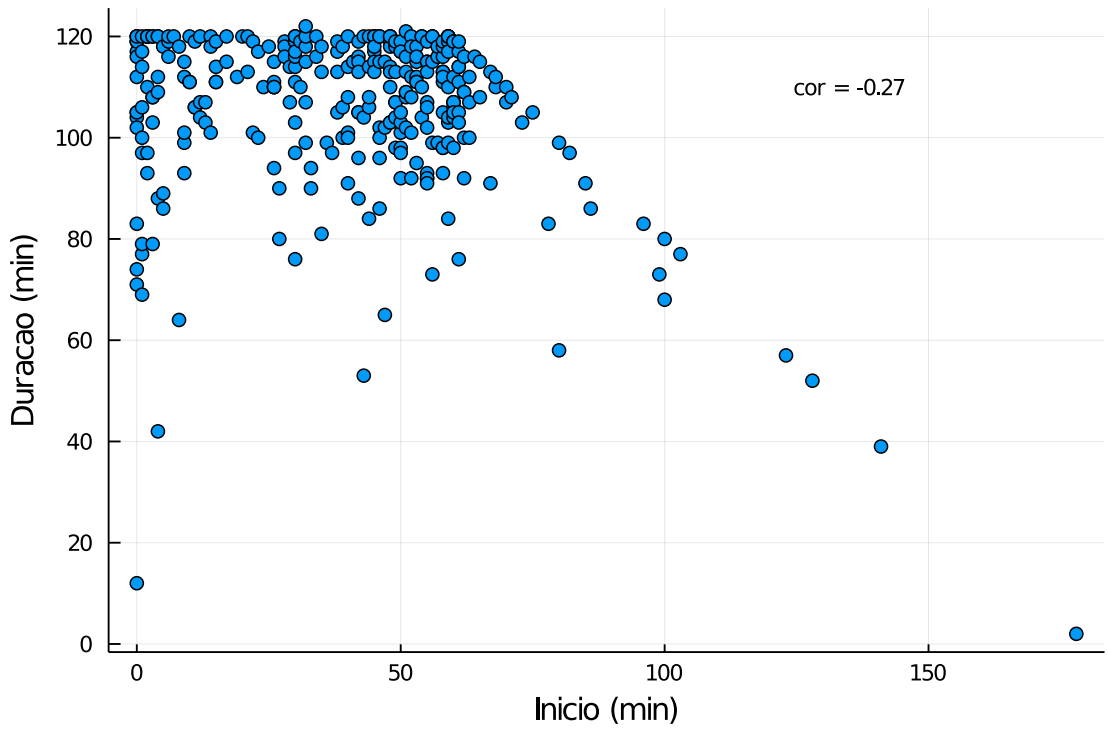
[22]:
```
scatter(df_P1.Inicio, df_P1.Termino, xlabel="Inicio (min)", ylabel="Termino␣
↪(min)", yticks=25:50:175, legend=false)
corout = cor(df_P1.Inicio, df_P1.Termino)
aux3=annotate!(130, 90, text("cor = "*string(round(corout,digits=2)), 8))
```

[22]:



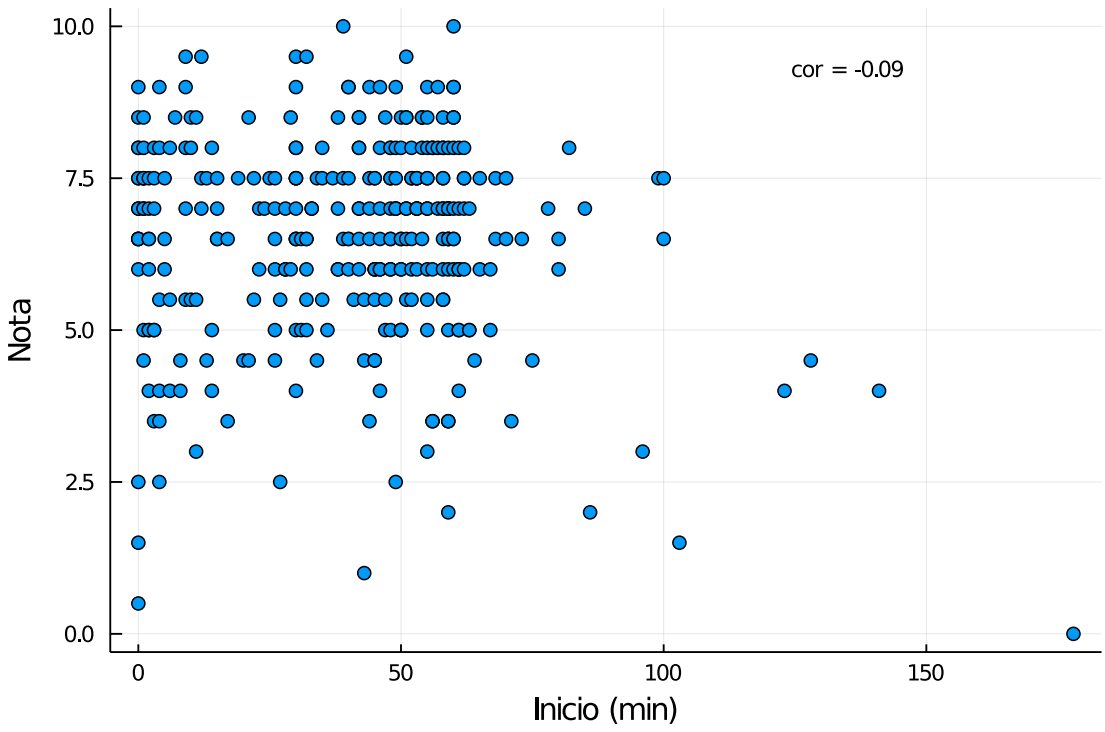
[23]:
```
scatter(df_P1.Inicio, df_P1.Duracao, xlabel="Inicio (min)", ylabel="Duracao␣
↪(min)", legend=false)
corout = cor(df_P1.Inicio, df_P1.Duracao)
aux4=annotate!(135, 110, text("cor = "*string(round(corout,digits=2)), 8))
```

[23]:



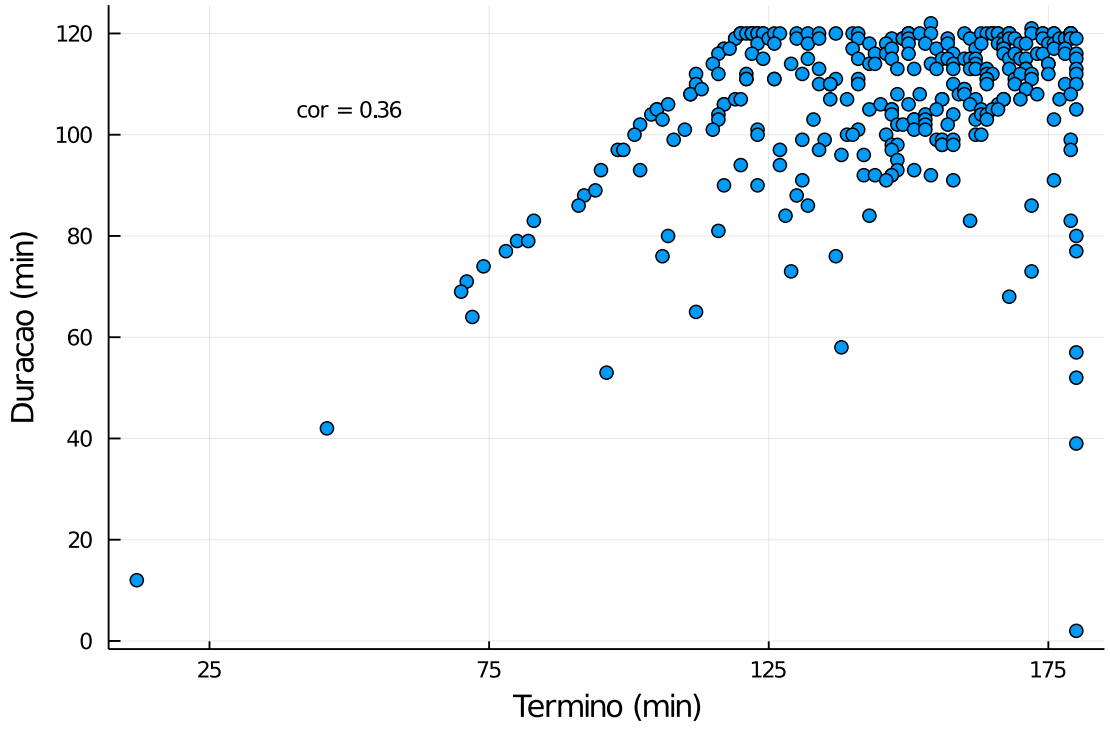
[24]:
```
scatter(df_P1.Inicio, df_P1.Nota, xlabel="Inicio (min)", ylabel="Nota",␣
↪legend=false)
corout = cor(df_P1.Inicio, df_P1.Nota)
aux5=annotate!(135, 9.3, text("cor = "*string(round(corout,digits=2)), 8))
```

[24]:



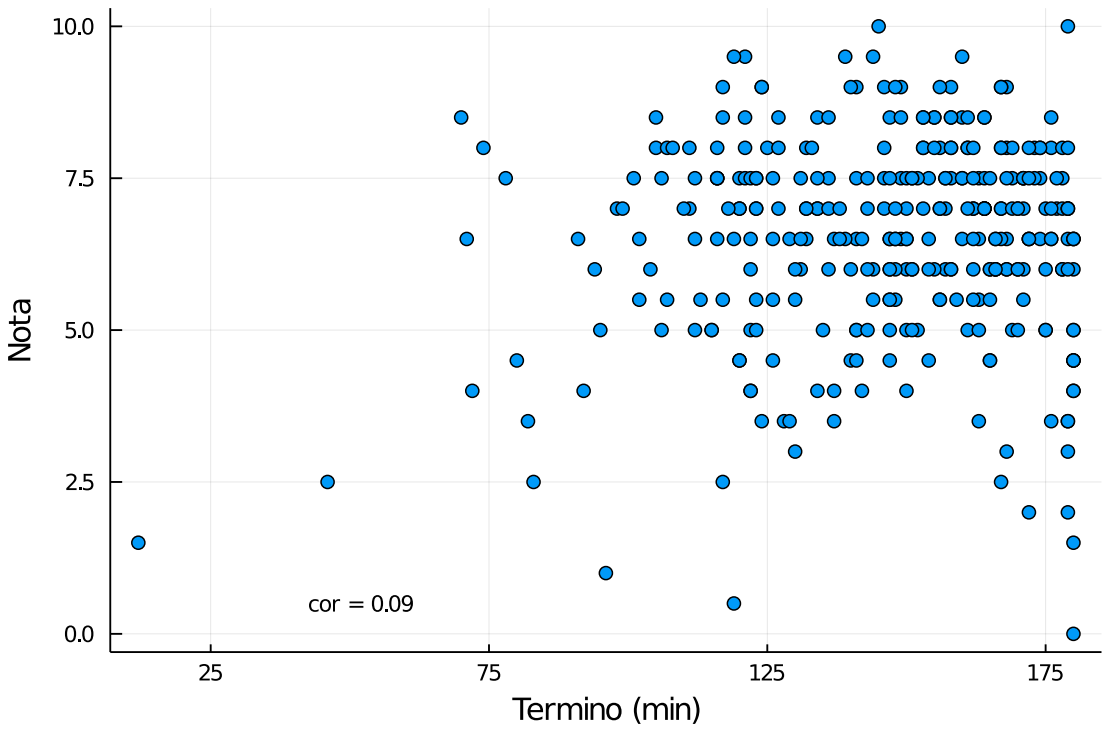
[25]:
```
scatter(df_P1.Termino, df_P1.Duracao, xlabel="Termino (min)", xticks=25:50:175,␣
↪ylabel="Duracao (min)", legend=false)
corout = cor(df_P1.Termino, df_P1.Duracao)
aux6=annotate!(50, 105, text("cor = "*string(round(corout,digits=2)), 8))
```

[25]:



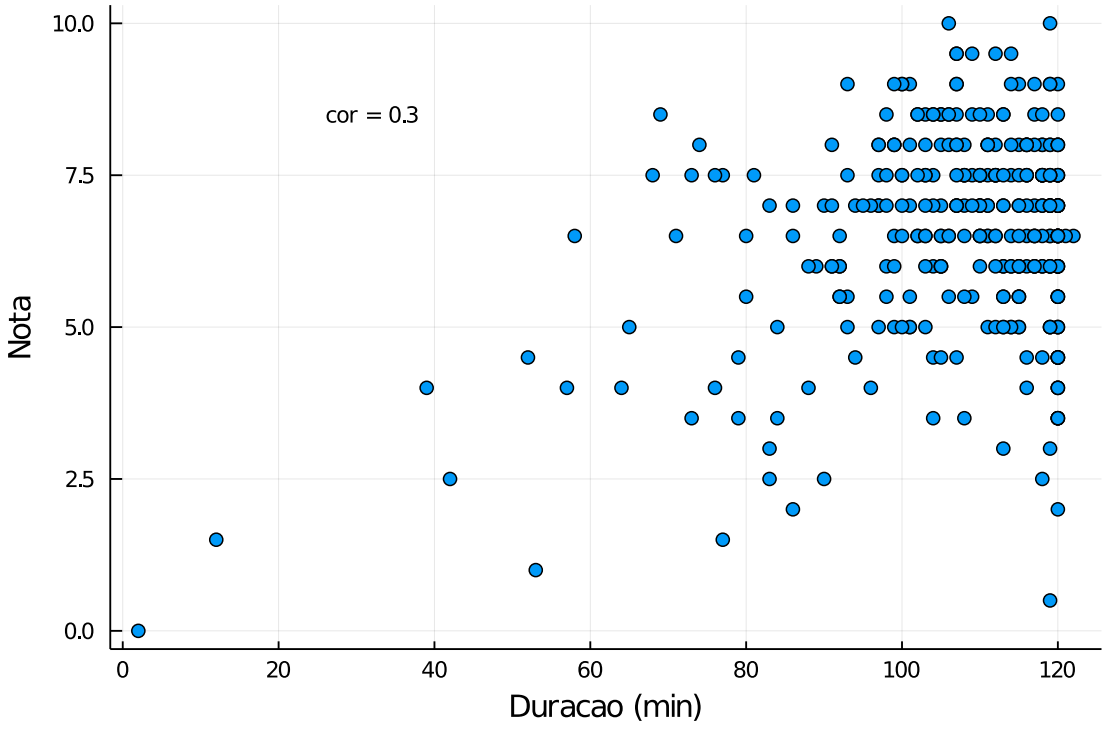
[26]:
```
scatter(df_P1.Termino, df_P1.Nota, xlabel="Termino (min)", xticks=25:50:175,␣
↪ylabel="Nota", legend=false)
corout = cor(df_P1.Termino, df_P1.Nota)
aux7=annotate!(52, 0.5, text("cor = "*string(round(corout,digits=2)), 8))
```

[26]:



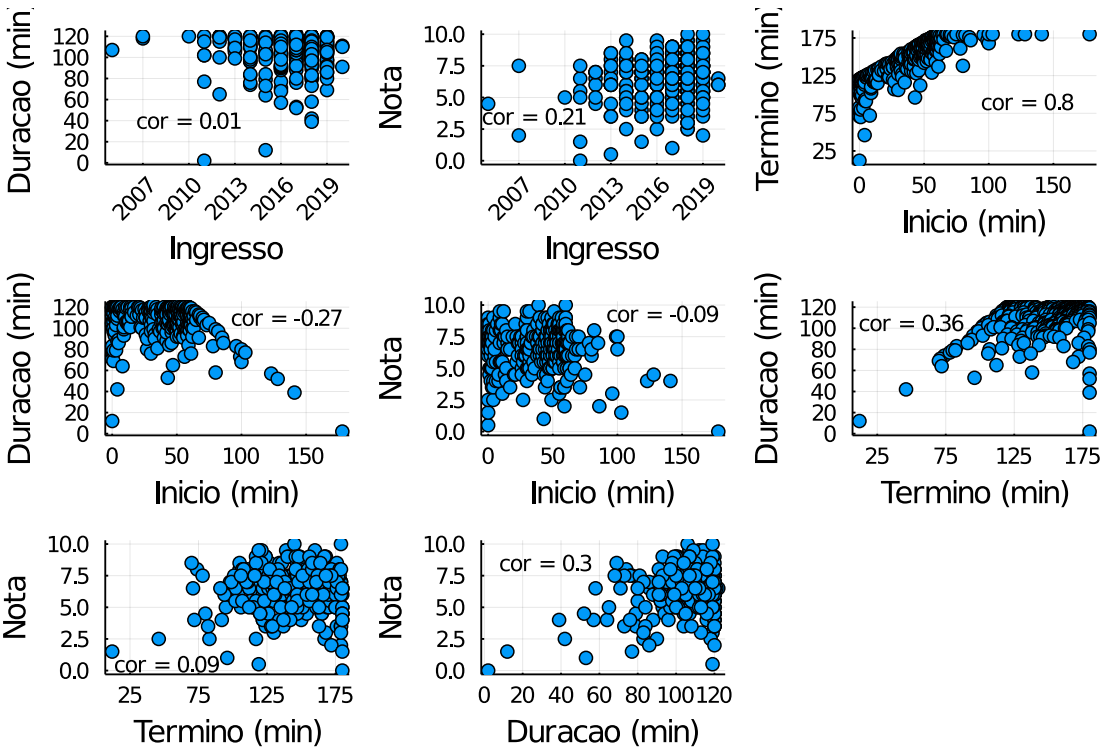
[27]:
```
scatter(df_P1.Duracao, df_P1.Nota, xlabel="Duracao (min)", ylabel="Nota",␣
↪legend=false)
corout = cor(df_P1.Duracao, df_P1.Nota)
aux8=annotate!(32, 8.5, text("cor = "*string(round(corout,digits=2)), 8))
```

[27]:



[28]:
```
plot(aux1,aux2, aux3, aux4, aux5, aux6, aux7, aux8)
```

[28]:



2.0.4 Dados de entrada relativos à Segunda Prova (P2)

[29]:
```
abrev_questionario_P2 = "P2"
caminho_completo_ficheiro_original_P2 = "P2_notas.csv"
N_questoes_P2 = 15

limite_duracao_P2 = 90
janela_Ini_P2 = "21 Feb 2021 15:30"
janela_Fin_P2 = "21 Feb 2021 18:00"
N_participantes_P2 = 323;
```

[30]:
```
dtformat = DateFormat("d u y H:M")
janela_ini_P2 = DateTime(janela_Ini_P2, dtformat)
janela_fin_P2 = DateTime(janela_Fin_P2, dtformat)
if !(janela_ini_P2 < janela_fin_P2)
    error("Duração da janela de acesso vazia!")
end
if (round(janela_fin_P2 - janela_ini_P2, Minute).value < limite_duracao_P2 < 0)
    error("Limite de duração do questionário menor que janela ou negativo!")
end
```

[31]:
```
df_P2 = DataFrame(CSV.File(caminho_completo_ficheiro_original_P2, type =␣
 ↪String))[1:end-1,:];
```

[32]:
```
if N_participantes_P2 != nrow(df_P2)
    error("Número declarado de participantes não coincide com número de linhas␣
 ↪menos um do ficheiro lido!")
elseif ncol(df_P2) != (N_questoes_P2 + 9)
    error("Número de questões mais nove não coincide com o número de colunas do␣
 ↪ficheiro lido!")
end
```

[33]:
```
names(df_P2)
select!(df_P2, Not([1, 2, 4, 9]));
```

[34]:
```
new_col_names_P2 = [Symbol("Q$i") for i=1:N_questoes_P2]
new_col_names_P2 = [Symbol("Ingresso"); Symbol("Inicio"); Symbol("Termino");
    Symbol("Duracao");  Symbol("Nota"); new_col_names_P2]
names!(df_P2, new_col_names_P2);
```

[35]:
```
df_P2 = replace.(df_P2, ","=>".")
df_P2 = replace.(df_P2, "-"=>"0.00");
```

[36]:
```
transform!(df_P2, :Ingresso => ByRow(x -> "20"*x[2:3]) => :Ingresso);
```

[37]:
```
df_P2 = replace.(df_P2, "janeiro"=>"Jan")
df_P2 = replace.(df_P2, "fevereiro"=>"Feb")
df_P2 = replace.(df_P2, "março"=>"Mar")
df_P2 = replace.(df_P2, "abril"=>"Apr")
df_P2 = replace.(df_P2, "maio"=>"May")
df_P2 = replace.(df_P2, "junho"=>"Jun")
df_P2 = replace.(df_P2, "julho"=>"Jul")
df_P2 = replace.(df_P2, "agosto"=>"Aug")
df_P2 = replace.(df_P2, "setembro"=>"Sep")
df_P2 = replace.(df_P2, "outubro"=>"Oct")
df_P2 = replace.(df_P2, "novembro"=>"Nov")
df_P2 = replace.(df_P2, "dezembro"=>"Dec")

dtformat = DateFormat("d u Y  H:M ")
df_P2.Inicio = DateTime.(df_P2.Inicio, dtformat)
df_P2.Termino = DateTime.(df_P2.Termino, dtformat)

transform!(df_P2, :Inicio => ByRow(x -> round(x - janela_ini_P2, Minute).value)␣
↪=> :Inicio)
transform!(df_P2, :Termino => ByRow(x -> round(x - janela_ini_P2, Minute).
↪value) => :Termino)
transform!(df_P2, [:Termino, :Inicio] => ByRow(-) => :Duracao);
```

[38]:
```
for i in 5:(N_questoes_P2 + 5)
    df_P2[names(df_P2)[i]] = parse.(Float64, df_P2[names(df_P2)[i]])
end
df_P2.Ingresso = parse.(Int, df_P2.Ingresso);
```

2.0.5 Estatísticas da P2

[39]:
```
describe(df_P2, :mean, :std, :min, :q25, :median, :q75, :max)
```

[39]:

	variable	mean	std	min	q25	median	q75	max
	Symbol	Float64	Float64	Real	Float64	Float64	Float64	Real
1	Ingresso	2016.5	2.32643	2005	2015.0	2017.0	2018.0	2020
2	Inicio	32.1207	22.4858	0	10.5	30.0	53.0	91
3	Termino	110.864	24.7993	26	92.0	113.0	131.0	150
4	Duracao	78.743	10.9169	26	72.0	81.0	88.0	92
5	Nota	7.17341	1.7604	0.66	6.0	7.33	8.66	10.0
6	Q1	0.555913	0.252229	0.0	0.67	0.67	0.67	0.67
7	Q2	0.427307	0.322532	0.0	0.0	0.67	0.67	0.67
8	Q3	0.557988	0.250391	0.0	0.67	0.67	0.67	0.67
9	Q4	0.501981	0.290868	0.0	0.335	0.67	0.67	0.67
10	Q5	0.485387	0.299812	0.0	0.0	0.67	0.67	0.67
11	Q6	0.510279	0.285929	0.0	0.67	0.67	0.67	0.67
12	Q7	0.406563	0.327775	0.0	0.0	0.67	0.67	0.67
13	Q8	0.400341	0.329076	0.0	0.0	0.67	0.67	0.67
14	Q9	0.406563	0.327775	0.0	0.0	0.67	0.67	0.67
15	Q10	0.487461	0.298759	0.0	0.0	0.67	0.67	0.67
16	Q11	0.492446	0.287694	0.0	0.0	0.66	0.66	0.66
17	Q12	0.461796	0.303009	0.0	0.0	0.66	0.66	0.66
18	Q13	0.541486	0.253718	0.0	0.66	0.66	0.66	0.66
19	Q14	0.437276	0.312561	0.0	0.0	0.66	0.66	0.66
20	Q15	0.500619	0.282908	0.0	0.66	0.66	0.66	0.66

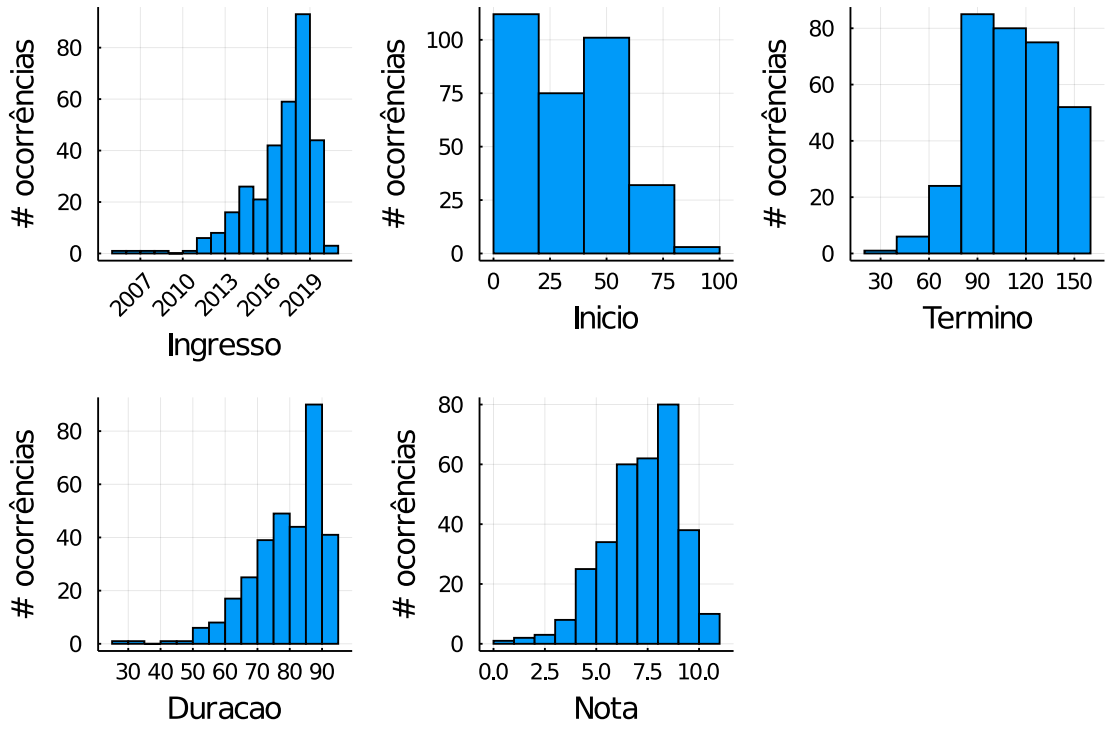
[40]:
```
modes.(eachcol(df_P2))
```

[40]:
```
20-element Array{Array{T,1} where T,1}:
 [2018]
 [0]
 [115]
 [90]
 [9.33]
 [0.67]
 [0.67]
 [0.67]
 [0.67]
 [0.67]
 [0.67]
 [0.67]
 [0.67]
 [0.67]
 [0.67]
 [0.66]
 [0.66]
 [0.66]
 [0.66]
 [0.66]
```

[41]:
```
p_P2 = []
for i in names(df_P2)
    if i == "Ingresso"
        push!(p_P2, histogram(df_P2[i], xlabel = i, ylabel = "# ocorrências",␣
 ↪legend=false, xrotation = 45))
    else
        push!(p_P2, histogram(df_P2[i], xlabel = i, ylabel = "# ocorrências",␣
 ↪legend=false))
    end
end
```

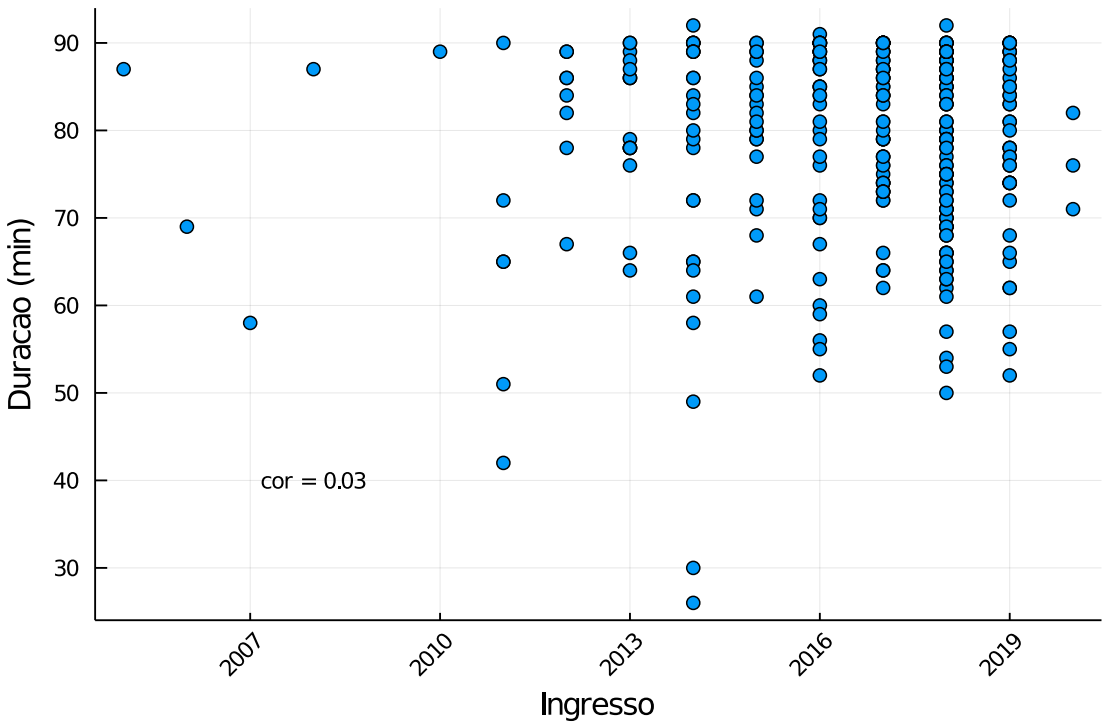
[42]:
```
plot(p_P2[1],p_P2[2],p_P2[3],p_P2[4],p_P2[5])
```

[42]:



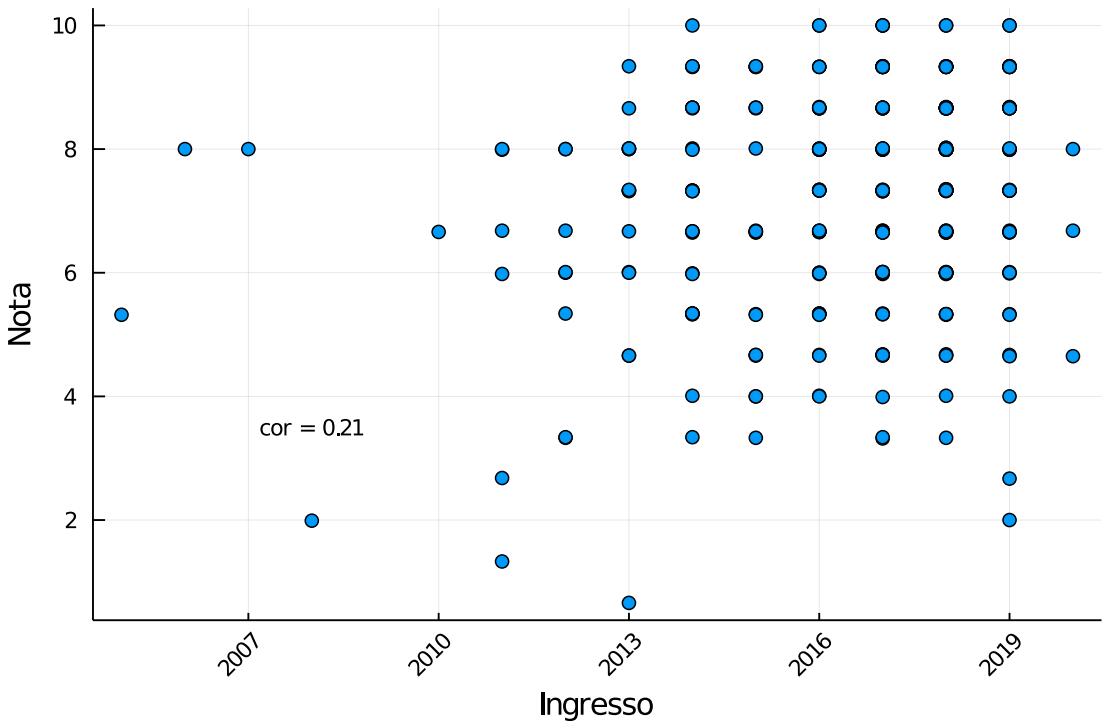
[43]:
```
scatter(df_P2.Ingresso, df_P2.Duracao, xlabel="Ingresso", ylabel="Duracao␣
 ↪(min)", xrotation=45, legend=false)
corout = cor(df_P2.Ingresso, df_P2.Duracao)
aux1=annotate!(2008, 40, text("cor = "*string(round(corout,digits=2)), 8))
```

[43]:



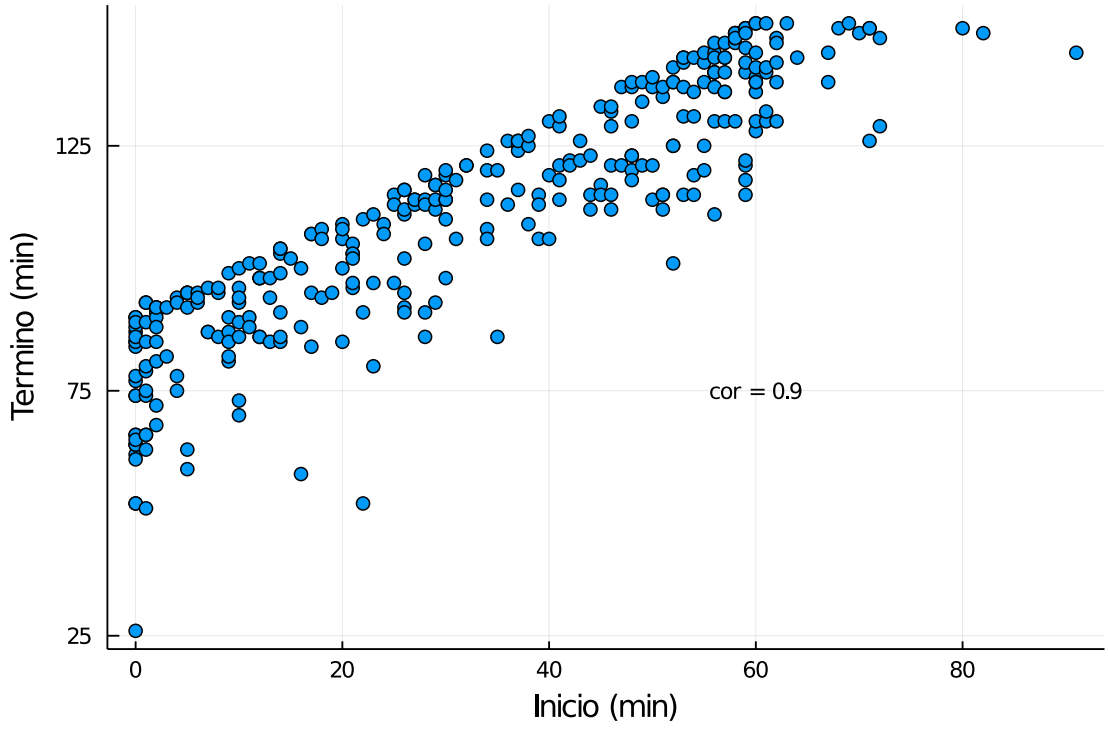
[44]:
```
scatter(df_P2.Ingresso, df_P2.Nota, xlabel="Ingresso", ylabel="Nota",␣
 ↪xrotation=45, legend=false)
corout = cor(df_P2.Ingresso, df_P2.Nota)
aux2=annotate!(2008, 3.5, text("cor = "*string(round(corout,digits=2)), 8))
```

[44]:



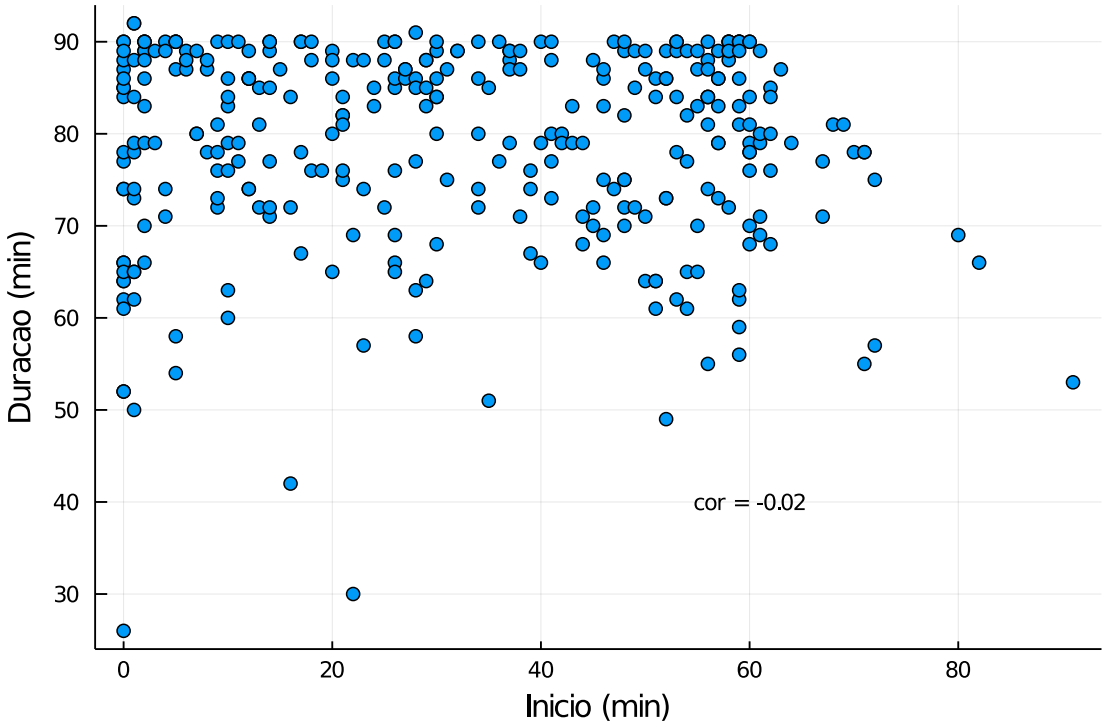
[45]:
```
scatter(df_P2.Inicio, df_P2.Termino, xlabel="Inicio (min)", ylabel="Termino␣
↪(min)", yticks=25:50:175, legend=false)
corout = cor(df_P2.Inicio, df_P2.Termino)
aux3=annotate!(60, 75, text("cor = "*string(round(corout,digits=2)), 8))
```

[45]:



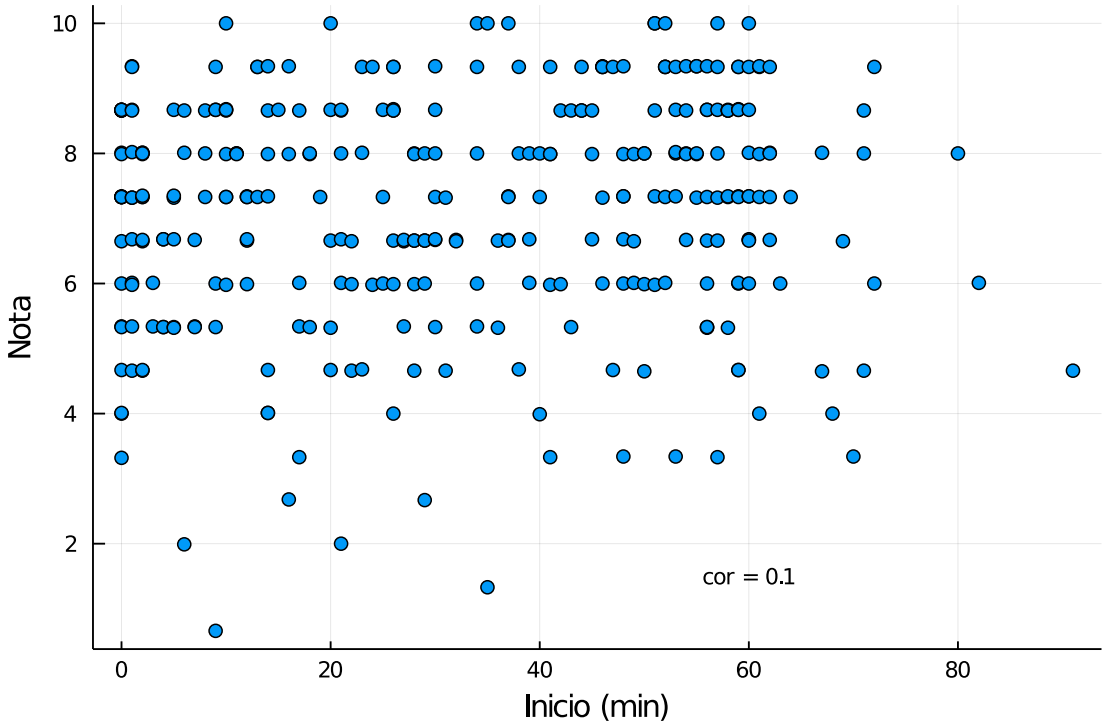
[46]:
```
scatter(df_P2.Inicio, df_P2.Duracao, xlabel="Inicio (min)", ylabel="Duracao␣
↪(min)", legend=false)
corout = cor(df_P2.Inicio, df_P2.Duracao)
aux4=annotate!(60, 40, text("cor = "*string(round(corout,digits=2)), 8))
```

[46]:



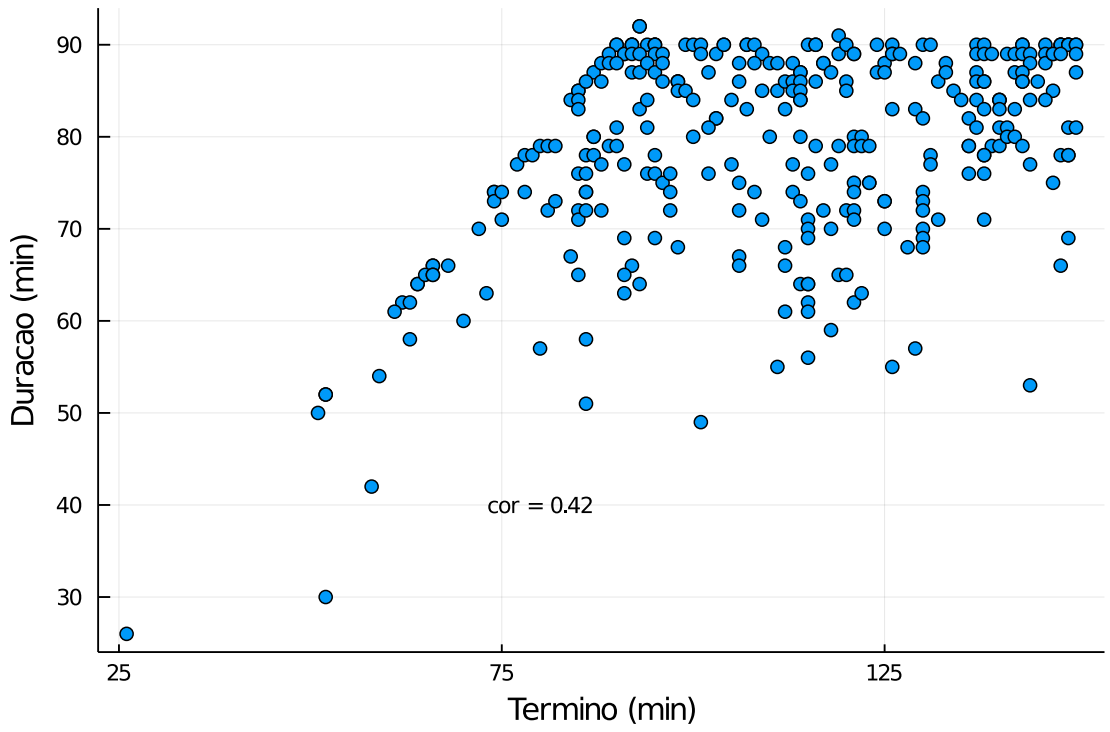
[47]:
```
scatter(df_P2.Inicio, df_P2.Nota, xlabel="Inicio (min)", ylabel="Nota",␣
↪legend=false)
corout = cor(df_P2.Inicio, df_P2.Nota)
aux5=annotate!(60, 1.5, text("cor = "*string(round(corout,digits=2)), 8))
```

[47]:



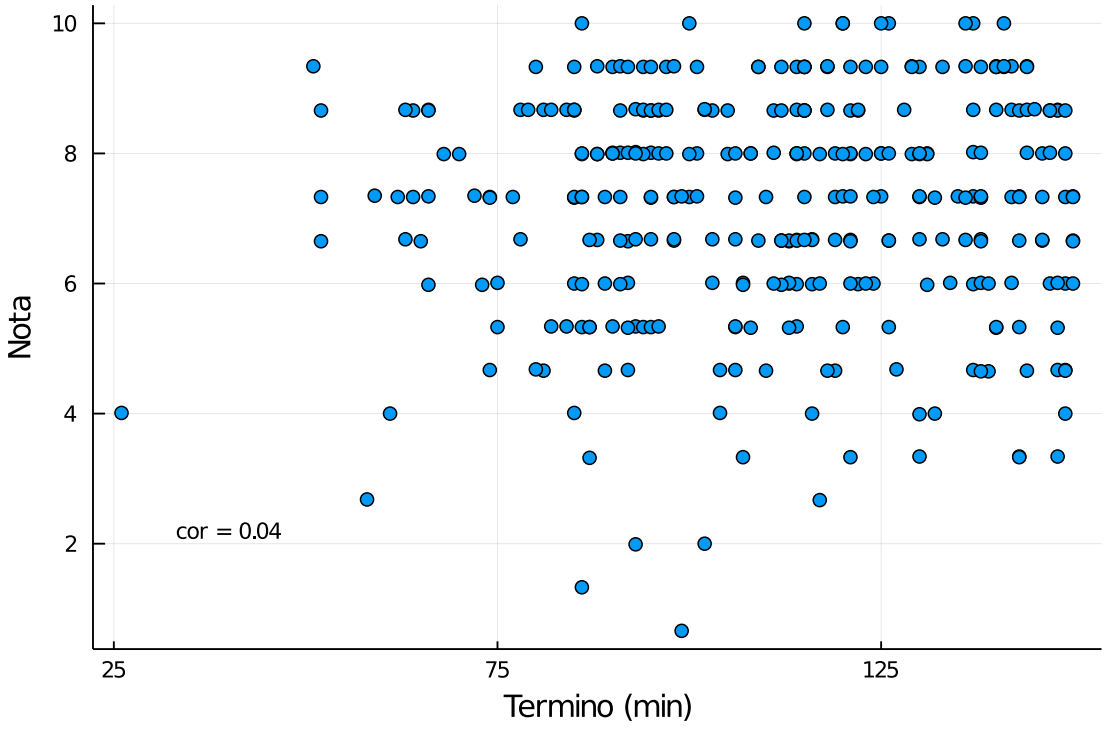
[48]:
```
scatter(df_P2.Termino, df_P2.Duracao, xlabel="Termino (min)", xticks=25:50:175,␣
↪ylabel="Duracao (min)", legend=false)
corout = cor(df_P2.Termino, df_P2.Duracao)
aux6=annotate!(80, 40, text("cor = "*string(round(corout,digits=2)), 8))
```

[48]:



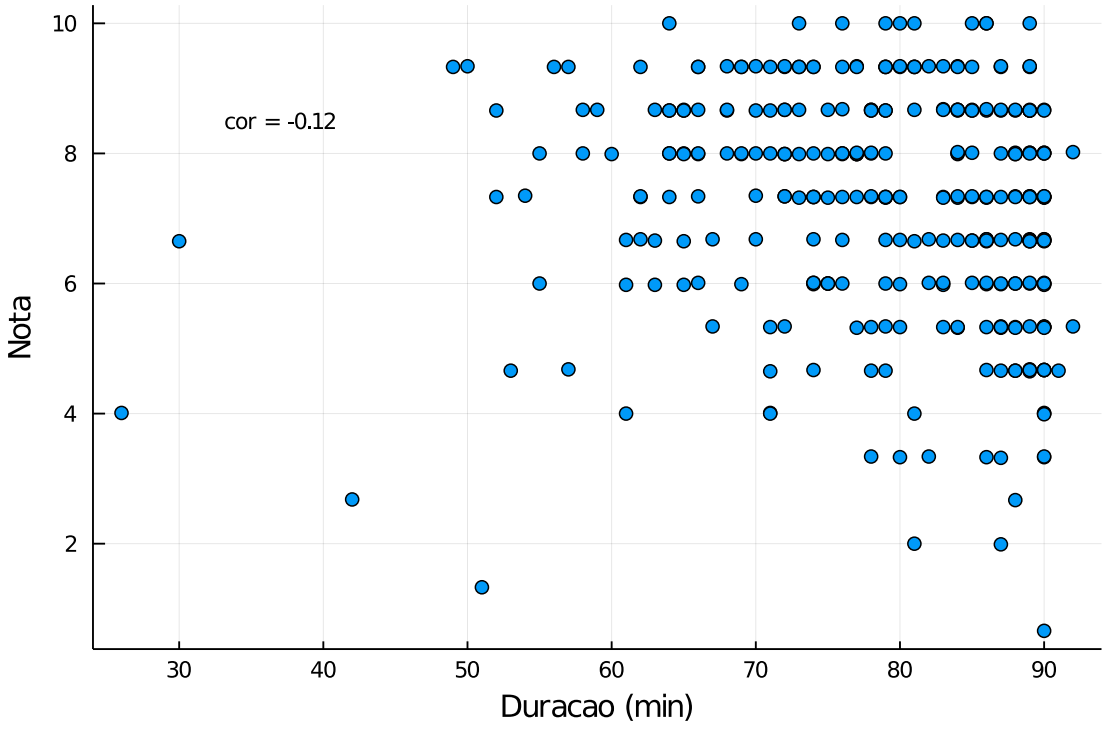
[49]:
```
scatter(df_P2.Termino, df_P2.Nota, xlabel="Termino (min)", xticks=25:50:175,␣
↪ylabel="Nota", legend=false)
corout = cor(df_P2.Termino, df_P2.Nota)
aux7=annotate!(40, 2.2, text("cor = "*string(round(corout,digits=2)), 8))
```

[49]:



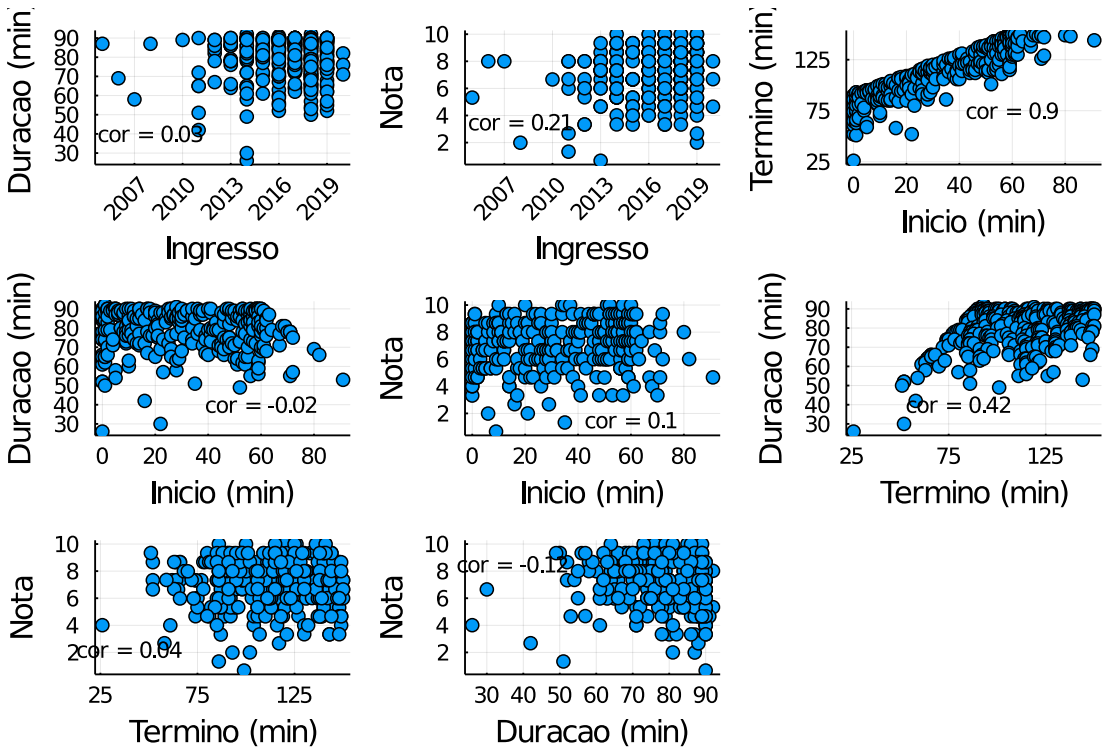
[50]:
```
scatter(df_P2.Duracao, df_P2.Nota, xlabel="Duracao (min)", ylabel="Nota",␣
↪legend=false)
corout = cor(df_P2.Duracao, df_P2.Nota)
aux8=annotate!(37, 8.5, text("cor = "*string(round(corout,digits=2)), 8))
```

[50]:



[51]:
```
plot(aux1,aux2, aux3, aux4, aux5, aux6, aux7, aux8)
```

[51]:



2.0.6 Estatísticas conjuntas da P1 e P2

[52]:
```
#ficheiro_original = "P1_P2_notas.csv"
#df = DataFrame(CSV.File(ficheiro_original, missingstring = "-"));
```

Aprovação direto

[53]:
```
#transform!(df, :, [:P1, :P2] => ByRow(+) => :Media_Parcial)
#df.Media_Parcial = round.(df.Media_Parcial/2,digits=1,RoundNearestTiesUp)
#n_alunos_fis3a = 386
#aprovacao_direto = length(filter(x -> !ismissing(x) && x > 7.0, df.
↪Media_Parcial))/n_alunos_fis3a
```

Correlação entre P1 e P2

[54]:
```
#comp_cases = dropmissing(df)
#cor(comp_cases.P1, comp_cases.P2), size(comp_cases)
```

[ ]:
```

```