

## See chapter 2 in Regression and Other Stories.

=====

Widen the notebook.

```
• html"""
• <style>
•   main {
•     margin: 0 auto;
•     max-width: 2000px;
•     padding-left: max(160px, 10%);
•     padding-right: max(160px, 10%);
•   }
• </style>
• """
```

```
• using Pkg ✓ , DrWatson ✓
```

A typical set of Julia packages to include in  
notebooks.

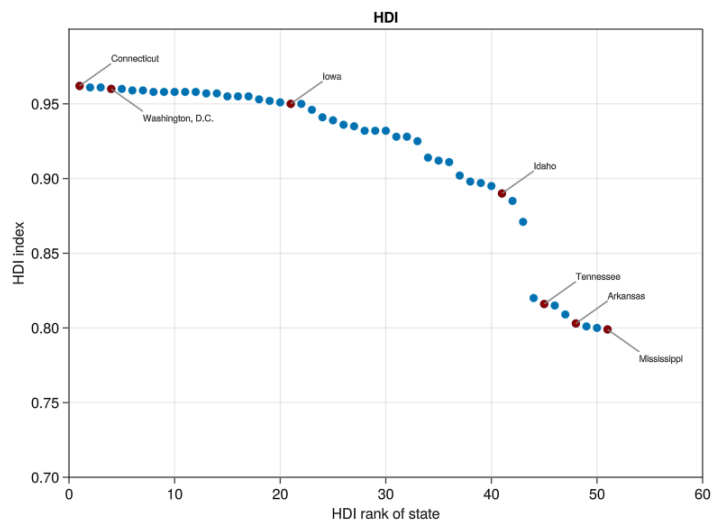
```
• begin
•   # Specific to this notebook
•   using GLM ✓
•
•   # Specific to ROSTuringPluto
•   using Optim ✓
•   using Logging ✓
•   using Turing ✓
•
•   # Graphics related
•   using GLMakie ✓
•
•   # Include basic packages
•   using RegressionAndOtherStories ✓
•   import RegressionAndOtherStories: link
•
•   Logging.disable_logging(Logging.Warn);
• end;
```

## 2.1 Examining where data come from.

`hdi =`

	rank	state	hdi	canada
<b>1</b>	1	"Connecticut"	0.962	2
<b>2</b>	2	"Massachusetts"	0.961	2
<b>3</b>	3	"New Jersey"	0.961	2
<b>4</b>	4	"Washington, D.C."	0.96	4
<b>5</b>	5	"Maryland"	0.96	3
<b>6</b>	6	"Hawaii"	0.959	2
<b>7</b>	7	"New York"	0.959	1
<b>8</b>	8	"New Hampshire"	0.958	1
<b>9</b>	9	"Minnesota"	0.958	1
<b>10</b>	10	"Rhode Island"	0.958	3
	⋮ more			
<b>51</b>	51	"Mississippi"	0.799	5

```
• hdi = CSV.read(ros_datadir("HDI",  
  "hdi.csv"), DataFrame)
```



```

let
  f = Figure()
  ax = Axis(f[1, 1]; title = "HDI",
    xlabel = "HDI rank of state", ylabel
    = "HDI index")
  limits!(ax, 0, 60, 0.7, 1)
  scatter!(hdi.rank, hdi.hdi)
  selection = 1:20:50
  scatter!(hdi.rank[selection],
    hdi.hdi[selection]; color=:darkred)
  for i in selection
    lines!([hdi.rank[i], hdi.rank[i] +
      3],
      [hdi.hdi[i], hdi.hdi[i] +
        0.015]; color=:grey)
    annotations!(hdi.state[i],
      position = (hdi.rank[i] + 3,
        hdi.hdi[i] + 0.015),
      fontsize = 10)
  end
  selection = [4, 51]
  scatter!(hdi.rank[selection],
    hdi.hdi[selection]; color=:darkred)
  for i in selection
    lines!([hdi.rank[i], hdi.rank[i] +
      3],
      [hdi.hdi[i], hdi.hdi[i] -
        0.015]; color=:grey)
    annotations!(hdi.state[i],
      position = (hdi.rank[i] + 3,
        hdi.hdi[i] - 0.023),
      fontsize = 10)
  end
  selection = 45:3:50
  scatter!(hdi.rank[selection],
    hdi.hdi[selection]; color=:darkred)

```

```

for i in selection
    lines!([hdi.rank[i], hdi.rank[i] +
3],
[hdi.hdi[i], hdi.hdi[i] +
0.015]; color=:grey)
    annotations!(hdi.state[i],
position = (hdi.rank[i] + 3,
hdi.hdi[i] + 0.015),
textsize = 10)
end
f
end

```

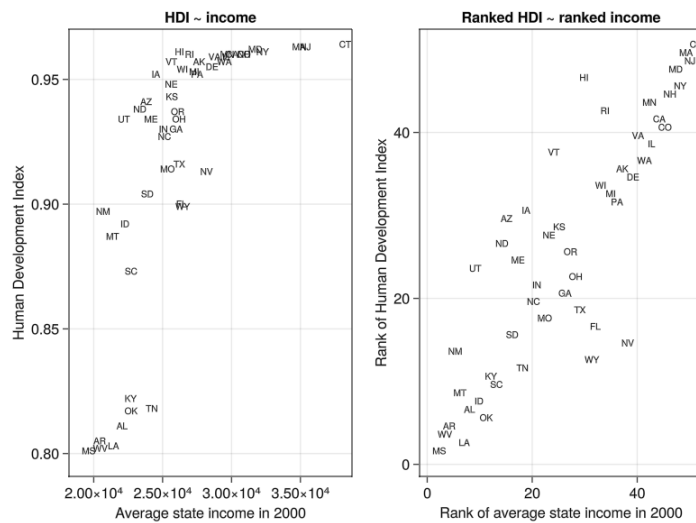
	st_state	st_stateabb	st_income
1	"Alabama"	"AL"	21656.2
2	"Alaska"	"AK"	27209.7
3	"Arizona"	"AZ"	23381.0
4	"Arkansas"	"AR"	19977.9
5	"California"	"CA"	29581.4
6	"Colorado"	"CO"	30406.0
7	"Connecticut"	"CT"	37808.2
8	"Delaware"	"DE"	28128.1
9	"Florida"	"FL"	25977.8
10	"Georgia"	"GA"	25502.2
	⋮ more		
50	"Wyoming"	"WY"	25934.1

```

• begin
•     votes = CSV.read(ros_datadir("HDI",
•         "votes.csv"), DataFrame;
•         delim=",", stringtype=String,
•         pool=false)
•     votes[votes.st_year .== 2000,
[:st_state, :st_stateabb, :st_income]]
end

```





```

let
.   f = Figure()
.   ax = Axis(f[1, 1]; title = "HDI ~
.   income",
.   xlabel = "Average state income in
.   2000",
.   ylabel = "Human Development Index")
.   for i in 1:size(hdivotes, 1)
.   if length(hdivotes.abbr[i]) > 0
.   annotations!(hdivotes.abbr[i],
.   position =
.   (hdivotes.income[i],
.   hdivotes.hdi[i]),
.   fontsize = 10)
.   end
.   end
.   hdivotes.rank_hdi =
.   sortperm(hdivotes.hdi)
.   global hdivotes2 = sort(hdivotes,
.   :income)
.   ax = Axis(f[1, 2]; title = "Ranked HDI
.   ~ ranked income",
.   xlabel = "Rank of average state
.   income in 2000",
.   ylabel = "Rank of Human Development
.   Index")
.   for i in 1:size(hdivotes2, 1)
.   if length(hdivotes2.abbr[i]) > 0
.   annotations!(hdivotes2.abbr[i],
.   position = (i,
.   hdivotes2.rank_hdi[i]),
.   fontsize = 10)
.   end
.   end
.   current_figure()
end

```

	rank	state	hdi	canada.dis
<b>1</b>	51	"Mississippi"	0.799	5
<b>2</b>	50	"West Virginia"	0.8	3
<b>3</b>	48	"Arkansas"	0.803	4
<b>4</b>	40	"New Mexico"	0.895	4
<b>5</b>	42	"Montana"	0.885	1
<b>6</b>	49	"Louisiana"	0.801	5
<b>7</b>	47	"Alabama"	0.809	5
<b>8</b>	29	"Utah"	0.932	2
<b>9</b>	41	"Idaho"	0.89	1
<b>10</b>	46	"Oklahoma"	0.815	4
	: more			
<b>50</b>	1	"Connecticut"	0.962	2

- [hdivotes2](#)

ppl2\_1 (generic function with 2 methods)

```

• @model function ppl2_1(inc, hdi)
•   a ~ Normal(0, 5)
•   b ~ Normal(0, 5)
•   σ ~ Exponential(1)
•   μ = a .+ b .* inc
•   for i in eachindex(hdi)
•     hdi[i] ~ Normal(μ[i], σ)
•   end
• end

```

	parameters	mean	std	naive_s
1	:a	3.05014	1.99214	0.031498
2	:b	0.875331	0.0684752	0.001082
3	: $\sigma$	7.22582	0.688018	0.010878

```

• begin
•   rank_income = collect(1:size(hdivotes2,
•     1))
•   rank_hdi = hdivotes2.rank_hdi
•   m2_1t = ppl2_1(rank_income, rank_hdi)
•   chns2_1t = sample(m2_1t, NUTS(),
•     MCMCThreads(), 1000, 4)
•   describe(chns2_1t)
end

```

	parameters	median	mad_sd	mean	st
1	"a"	3.055	1.945	3.05	1.99
2	"b"	0.873	0.066	0.875	0.06
3	" $\sigma$ "	7.154	0.65	7.226	0.68

```

• begin
•   post2_1t = DataFrame(chns2_1t)
•   ms2_1t = model_summary(post2_1t, [:a,
•     :b, : $\sigma$ ])
end

```

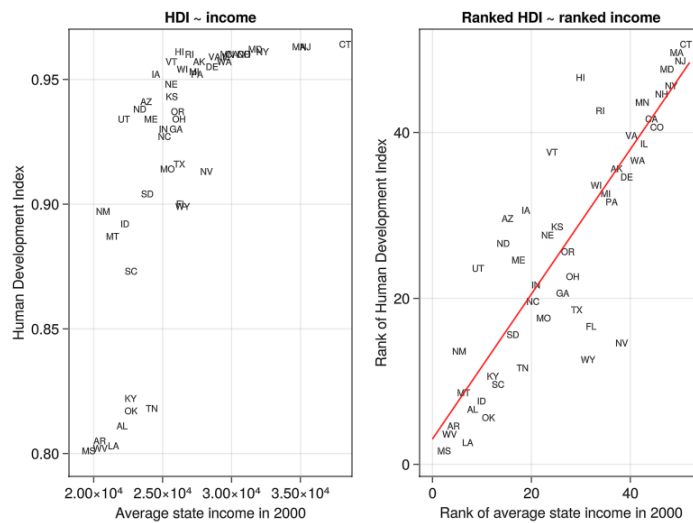
► [3.055, 0.873, 7.154]

```

•  $\bar{a}$ ,  $\bar{b}$ ,  $\bar{\sigma}$  = [ms2_1t[p, :median] for p in [:a,
•   :b, : $\sigma$ ]]

```





```

let
.   f = Figure()
.   ax = Axis(f[1, 1]; title = "HDI ~
.   income",
.   xlabel = "Average state income in
.   2000",
.   ylabel = "Human Development Index")
.   for i in 1:size(hdivotes, 1)
.   if length(hdivotes.abbr[i]) > 0
.   annotations!(hdivotes.abbr[i],
.   position =
.   (hdivotes.income[i],
.   hdivotes.hdi[i]),
.   fontsize = 10)
.   end
.   end
.   ax = Axis(f[1, 2]; title = "Ranked HDI
.   ~ ranked income",
.   xlabel = "Rank of average state
.   income in 2000",
.   ylabel = "Rank of Human Development
.   Index")
.   for i in 1:size(hdivotes2, 1)
.   if length(hdivotes2.abbr[i]) > 0
.   annotations!(hdivotes2.abbr[i],
.   position = (i,
.   hdivotes2.rank_hdi[i]),
.   fontsize = 10)
.   end
.   end
.   x = 0:52
.   lines!(x, ā .+ b̄ .* x; color=:red)
.   f
end

```

## 2.2 Validity and reliability.

	survey	regicert
1	"june08voter"	"absolutely certain"
2	"aug08relig"	"absolutely certain"
3	"aug08relig"	"absolutely certain"
4	"aug08relig"	"absolutely certain"
5	"june08voter"	"absolutely certain"
6	"july08poli-econ"	"absolutely certain"
7	"june08voter"	"absolutely certain"
8	"aug08relig"	"absolutely certain"
9	"june08voter"	"absolutely certain"
10	"july08poli-econ"	"absolutely certain"
: more		
31201	"sept08forpoli"	"absolutely certain"

```
• begin
•   pew_pre_raw =
•     CSV.read(ros_datadir("Pew", "pew.csv"),
•     DataFrame; missingstring="NA",
•     pool=false)
•   pew_pre = pew_pre_raw[:, [:survey,
•   :regicert, :party, :state, :heat2,
•   :heat4, :income2, :party4, :date,
•       :weight, :voter_weight2, :pid,
•       :ideology, :inc]]
• end
```

pid\_incprob =

	Column1	V1	V2	V3	
1	1	0.0	0.0	0.0	0.0
2	2	0.434068	0.410585	0.40081	0.3
3	3	0.583262	0.54593	0.51899	0.5
4	4	0.780714	0.717176	0.67472	0.6
5	5	0.849178	0.818929	0.778284	0.7
6	6	1.0	1.0	1.0	1.0

```
• pid_incprob = CSV.read(ros_datadir("Pew",  
  "pid_incprop.csv"), DataFrame;  
  missingstring="NA", pool=false)
```

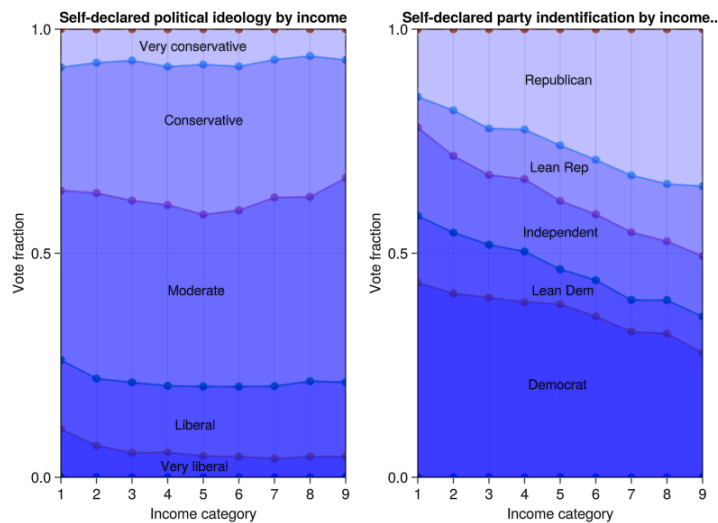
ideo\_incprob =

	Column1	V1	V2	V3	
1	1	0.0	0.0	0.0	0
2	2	0.10736	0.0713431	0.0545858	0
3	3	0.261838	0.220291	0.211839	0
4	4	0.640184	0.634329	0.617531	0
5	5	0.914783	0.925208	0.930161	0
6	6	1.0	1.0	1.0	1

```
• ideo_incprob = CSV.read(ros_datadir("Pew",  
  "ideo_incprop.csv"), DataFrame;  
  missingstring="NA", pool=false)
```

```
3x3 Matrix{Float64}:  
 0.00984338  0.00647884  0.00217881  
 0.00367166  0.0158111  0.00531952  
 0.00075453  0.00769068  0.0152036
```

```
• begin  
•   party_incprob_df =  
•   CSV.read(ros_datadir("Pew",  
•   "party_incprop.csv"), DataFrame;  
•   missingstring="NA", pool=false)  
•   party_incprob =  
•   reshape(Array(party_incprob_df)[:,  
•   2:end], :, 3, 9)  
•   party_incprob[:, :, 9]  
end
```



```

let
  x1 = 1.0:1.0:9.0
  f = Figure()
  ax = Axis(f[1, 1], title = "Self-
  declared political ideology by income",
    xlabel = "Income category", ylabel
    = "Vote fraction")
  limits!(ax, 1, 9, 0, 1)
  for i in 1:6
    sca1 = scatter!(x1,
      Array(ideo_incprob[i, 2:end]))
    lin = lines!(x1,
      Array(ideo_incprob[i, 2:end]))
    band!(x1, fill(0, length(x1)),
      Array(ideo_incprob[i, 2:end]);
      color = (:blue, 0.25), label =
      "Label")
  end
  annotations!("Very conservative",
    position = (3.2, 0.945), fontsize=15)
  annotations!("Conservative", position =
    (3.9, 0.78), fontsize=15)
  annotations!("Moderate", position =
    (4.0, 0.4), fontsize=15)
  annotations!("Liberal", position =
    (4.2, 0.1), fontsize=15)
  annotations!("Very liberal", position =
    (3.8, 0.0075), fontsize=15)
  ax = Axis(f[1, 2], title = "Self-
  declared party indentification by
  income..",
    xlabel = "Income category", ylabel
    = "Vote fraction")
  limits!(ax, 1, 9, 0, 1)
  for i in 1:6

```

```

        sca1 = scatter!(x1,
            Array(pid_incprob[i, 2:end]))
        lin = lines!(x1,
            Array(pid_incprob[i, 2:end]))
        band!(x1, fill(0, length(x1)),
            Array(pid_incprob[i, 2:end]));
            color = (:blue, 0.25), label =
                "Label")
    end
    annotations!("Republican", position =
        (4.0, 0.87), fontsize=15)
    annotations!("Lean Rep", position =
        (4.15, 0.675), fontsize=15)
    annotations!("Independent", position =
        (3.95, 0.53), fontsize=15)
    annotations!("Lean Dem", position =
        (4.2, 0.4), fontsize=15)
    annotations!("Democrat", position =
        (4.1, 0.19), fontsize=15)
    current_figure()
end

```

## 2.3 All graphs are comparisons.

health =	country	spending	lifespan
1	"Australia"	3357	81.4
2	"Austria"	3763	80.1
3	"Belgium"	3595	79.8
4	"Canada"	3895	80.7
5	"Czech"	1626	77.0
6	"Denmark"	3512	78.4
7	"Finland"	2840	79.5
8	"France"	3601	81.0
9	"Germany"	3588	80.0
10	"Greece"	2727	79.6
: more			
30	"USA"	7290	78.1

```

• health =
  CSV.read(ros_datadir("HealthExpenditure",
    "healthdata.csv"), DataFrame;
    missingstring="NA", pool=false)

```

ppl2\_3 (generic function with 2 methods)

```

• @model function ppl2_3(x, y)
•   a ~ Normal(0, 50)
•   b ~ Normal(0, 50)
•   σ ~ Exponential(1)
•   μ = a .+ b .* x
•   for i in eachindex(y)
•     y[i] ~ Normal(μ[i], σ)
•   end
• end

```

	parameters	mean	std	na:
1	:a	76.1548	1.16359	0.00
2	:b	0.00099446	0.000346675	5.48
3	: $\sigma$	2.17243	0.291672	0.00

```

• begin
•   m2_3t = ppl2_3(health.spending,
•                 health.lifespan)
•   chns2_3t = sample(m2_3t, NUTS(),
•                     MCMCThreads(), 1000, 4)
•   describe(chns2_3t)
end

```

	parameters	median	mad_sd	mean	st
1	"a"	76.168	1.088	76.155	1.16
2	"b"	0.001	0.0	0.001	0.0
3	" $\sigma$ "	2.134	0.27	2.172	0.29

```

• begin
•   post2_3t = DataFrame(chns2_3t)[: , 3:5]
•   ms2_3t = model_summary(post2_3t, [:a,
•                                     :b, : $\sigma$ ])
end

```

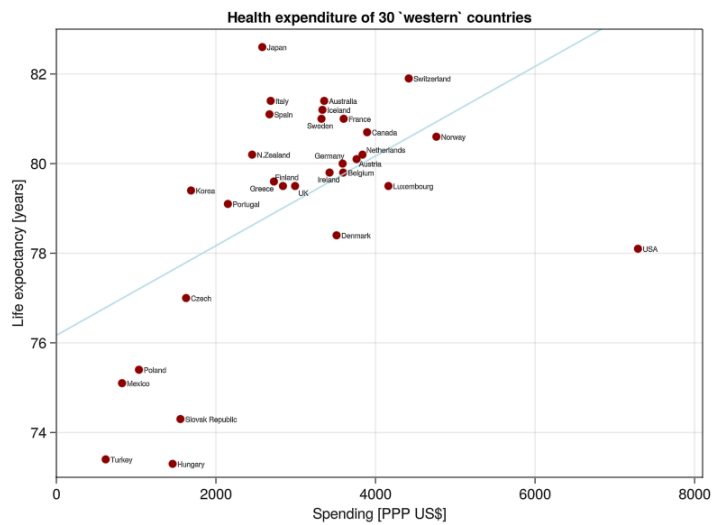
► [76.168, 0.001, 2.134]

```

• â, b̂, σ̂ = ms2_3t[: , :median]

```





```

let
  x = 0:8000
  f = Figure()
  ax = Axis(f[1, 1], title = "Health
expenditure of 30 `western` countries",
    xlabel = "Spending [PPP US$]",
    ylabel = "Life expectancy [years]")
  limits!(ax, 0, 8100, 73, 83)
  sca = scatter!(health.spending,
health.lifespan; color=:darkred)
  lin = lines!(x, â .+ b̂ * x;
color=:lightblue)
  for i in 1:nrow(health)
    if health.country[i] == "UK"
      annotations!(health.country[i],
        position =
          (health.spending[i]+40,
            health.lifespan[i]-0.25),
            fontsize=8)
    elseif health.country[i] ==
      "Finland"
      annotations!(health.country[i],
        position =
          (health.spending[i]-100,
            health.lifespan[i]+0.1),
            fontsize=8)
    elseif health.country[i] == "Greece"
      annotations!(health.country[i],
        position =
          (health.spending[i]-300,
            health.lifespan[i]-0.25),
            fontsize=8)
    elseif health.country[i] == "Sweden"
      annotations!(health.country[i],
        position =
          (health.spending[i]-180,

```

```

•         health.lifespan[i]-0.25),
•         textsize=8)
•     elseif health.country[i] ==
•     "Ireland"
        annotations!(health.country[i],
        position =
        (health.spending[i]-150,
        health.lifespan[i]-0.25),
        textsize=8)
    elseif health.country[i] ==
    "Netherlands"
        annotations!(health.country[i],
        position =
        (health.spending[i]+50,
        health.lifespan[i]+0.01),
        textsize=8)
    elseif health.country[i] ==
    "Germany"
        annotations!(health.country[i],
        position =
        (health.spending[i]-350,
        health.lifespan[i]+0.08),
        textsize=8)
    elseif health.country[i] ==
    "Austria"
        annotations!(health.country[i],
        position =
        (health.spending[i]+30,
        health.lifespan[i]-0.2),
        textsize=8)
    else
        annotations!(health.country[i],
        position =
        (health.spending[i]+60,
        health.lifespan[i]-0.1),
        textsize=8)
    end
end
current_figure()
end

```

**Names example.**

```
cleannames =
```

	X	name	sex	X1880	X1
1	1	"Mary"	"F"	7065	69
2	2	"Anna"	"F"	2604	26
3	3	"Emma"	"F"	2003	20
4	4	"Elizabeth"	"F"	1939	18
5	5	"Minnie"	"F"	1746	16
6	6	"Margaret"	"F"	1578	16
7	7	"Ida"	"F"	1472	14
8	8	"Alice"	"F"	1414	13
9	9	"Bertha"	"F"	1320	13
10	10	"Sarah"	"F"	1288	12
⋮ more					
98012	98148	"Zzyzx"	"M"	0	0

```
• cleannames = CSV.read(ros_datadir("Names",  
  "allnames_clean.csv"), DataFrame)
```

```
► (98012, 134)
```

```
• size(cleannames)
```

```
► ["X", "name", "sex", "X1880", "X1881", "X1882",
```

```
• names(cleannames)
```

df =

	name	sex	X1906	X1956	X2006
1	"John"	"M"	8263	80735	1514
2	"William"	"M"	6567	58927	1891
3	"James"	"M"	5908	84840	1621
4	"Charles"	"M"	3607	35198	7999
5	"George"	"M"	4201	17228	2699
6	"Frank"	"M"	2798	11126	1399
7	"Joseph"	"M"	3527	32706	1839
8	"Thomas"	"M"	2177	44785	9493
9	"Henry"	"M"	2111	5951	4661
10	"Robert"	"M"	3636	83869	9874
: more					
36659	"Zzyzx"	"M"	0	0	0

- `df = cleannames[cleannames.sex .== "M",  
["name", "sex", "X1906", "X1956", "X2006"]]`

- `letters = 'a':'z';`

count\_letters (generic function with 1 method)

```
• function count_letters(df::DataFrame,  
• years::Vector{String})  
•     letter_counts = DataFrame()  
•     for year in Symbol.(years)  
•  
•         !(year in Symbol.(names(df))) &&  
•         begin  
•             @warn "The year $(year) is not  
•                 present in df."  
•             continue  
•         end  
•  
•         tmpdf = df[:, [:name, year]]  
•  
•         yrcounts = zeros{Int,  
•             length(letters)}  
•         for (ind, letter) in  
•             enumerate(letters)  
•             yrcounts[ind] = sum(filter(row -  
•                 > row.name[end] == letter,  
•                 tmpdf)[: , 2])  
•         end  
•         letter_counts[!, year] = 100 *  
•             yrcounts / sum(yrcounts)  
•     end  
•     letter_counts  
end
```

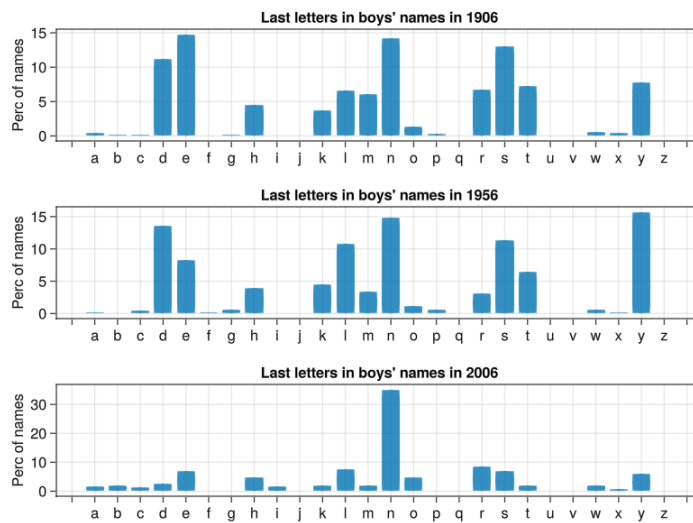
```
letter_count =
```

	X1906	X1956	X2006
<b>1</b>	0.473516	0.201557	1.76002
<b>2</b>	0.208106	0.0938356	2.07864
<b>3</b>	0.237512	0.466714	1.29964
<b>4</b>	11.2287	13.5269	2.50852
<b>5</b>	14.7936	8.31948	6.97453
<b>6</b>	0.104807	0.13137	0.0960275
<b>7</b>	0.200566	0.562255	0.0941245
<b>8</b>	4.5082	3.9746	4.79937
<b>9</b>	0.0806786	0.054927	1.63744
<b>10</b>	0.0	0.0	0.0605539
: more			
<b>26</b>	0.0180961	0.0176297	0.15058

```
• letter_count = count_letters(df, ["X1906",  
  "X1956", "X2006"])
```

```
► [100.0, 100.0, 100.0]
```

```
• sum.(eachcol(letter_count))
```



```

• let
•   f = Figure()
•   for (ind, year) in enumerate(["X1906",
•   "X1956", "X2006"])
•       ax = Axis(f[ind, 1], title="Last
•       letters in boys' names in
•       $(year[2:end])",
•       ylabel="Perc of names")
•       ax.xticks = (0:27, vcat(" ", string.
•       (letters), " "))
•       barplot!(f[ind, 1], 1:26,
•       letter_count[:, Symbol(year)],
•       width=0.8, gap=0.01)
•   end
•   f
• end

```

```
all_letter_count =
```

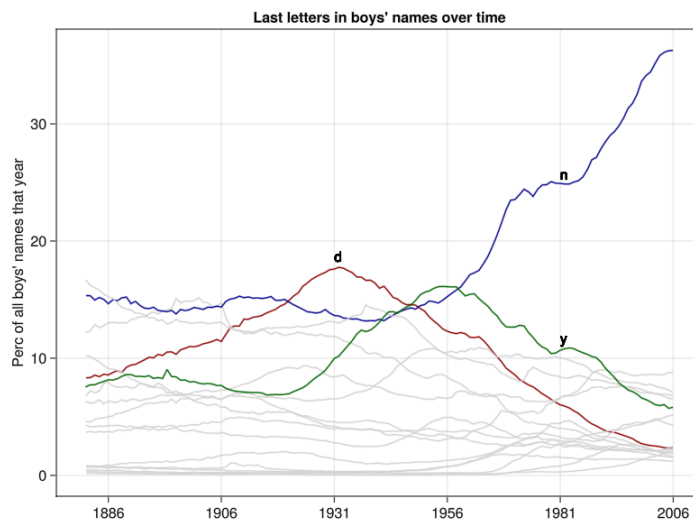
	X1880	X1881	X1882	X1883
1	0.683795	0.738044	0.675187	0.703254
2	0.461607	0.467494	0.446011	0.433066
3	0.316504	0.329235	0.307625	0.286475
4	8.32253	8.34229	8.55178	8.42755
5	12.2122	12.3229	12.855	12.602
6	0.0979441	0.0825575	0.118995	0.10156
7	0.133313	0.145222	0.128691	0.129345
8	3.66383	3.74095	3.66858	3.74142
9	0.181378	0.204902	0.182459	0.160005
10	0.0	0.0	0.0	0.0
⋮	more			
26	0.0262998	0.00795735	0.0273248	0.011497

```
• all_letter_count =  
  count_letters(cleannames[cleannames.sex ==  
    "M", :], names(cleannames[:, vcat(4:end)]))
```

```
► [0.473516, 0.208106, 0.237512, 11.2287, 14.7936
```

```
• all_letter_count[:, "X1906"]
```





```

let
  f = Figure()
  ax = Axis(f[1, 1], title="Last letters
in boys' names over time",
  ylabel="Perc of all boys' names
that year")
  ax.xticks = (6:25:131, ["1886", "1906",
"1931", "1956", "1981", "2006"])

  for l in 1:length(letters)
    col = :lightgrey
    if letters[l] == 'n'
      col = :darkblue
    elseif letters[l] == 'd'
      col = :darkred
    elseif letters[l] == 'y'
      col = :darkgreen
    end
    if maximum(Array(all_letter_count)
[l,:]) > 1
      lines!(1:size(all_letter_count,
2), Array(all_letter_count)
[l,:], color=col)
    end
    annotations!("n", position = (106,
25), textsize=15)
    annotations!("d", position = (56,
18), textsize=15)
    annotations!("y", position = (106,
11), textsize=15)
  end
  current_figure()
end

```

## 2.4 Data and adjustment.

Not yet done. A good exercise ...?

age =

	MONTH	YEAR	AGE	TOT_POP	TOT_M
1	4	2000	999	281424600	138056
2	4	2000	0	3805718	194905
3	4	2000	1	3820647	195313
4	4	2000	2	3790534	193903
5	4	2000	3	3832855	195899
6	4	2000	4	3926400	201069
7	4	2000	5	3965175	203110
8	4	2000	6	4019782	205826
9	4	2000	7	4118211	210989
10	4	2000	8	4179294	213787
⋮ more					
1131	7	2010	85	5532756	180716

```
• age =  
  CSV.read(ros_datadir("AgePeriodCohort", "US-  
EST00INT-ALLDATA.csv"), DataFrame)
```

```
► ["MONTH", "YEAR", "AGE", "TOT_POP", "TOT_MALE",
```

```
• names(age)
```

`white_nonhisp =`

	Age	Year	Deaths	Population	Rate
<b>1</b>	45	1999	8304	3166393	262.3
<b>2</b>	45	2000	8604	3207271	268.3
<b>3</b>	45	2001	8836	3152637	280.3
<b>4</b>	45	2002	9217	3256317	283.0
<b>5</b>	45	2003	9287	3260376	284.8
<b>6</b>	45	2004	9210	3211340	286.8
<b>7</b>	45	2005	9352	3279109	285.2
<b>8</b>	45	2006	9100	3222835	282.4
<b>9</b>	45	2007	8805	3137876	280.6
<b>10</b>	45	2008	8751	3074171	284.7
: more					
<b>150</b>	54	2013	17989	3141010	572.7

```
• white_nonhisp =  
  CSV.read(ros_datadir("AgePeriodCohort",  
    "white_nonhisp_death_rates.txt"),  
    DataFrame; delim="\t")
```

white\_hisp =

	Age	Male	Year	Deaths	Population
1	35	0	1999	1291	1578829
2	35	0	2000	1264	1528463
3	35	0	2001	1186	1377466
4	35	0	2002	1194	1333639
5	35	0	2003	1166	1302188
6	35	0	2004	1166	1325435
7	35	0	2005	1201	1383427
8	35	0	2006	1197	1349975
9	35	0	2007	1147	1239648
10	35	0	2008	1098	1155005
: more					
900	64	1	2013	17761	1255101

```
• white_hisp =  
  CSV.read(ros_datadir("AgePeriodCohort",  
    "white_nonhisp_death_rates_by_sex.txt"),  
    DataFrame; delim="\t")
```

white\_nonhisp\_by\_sex =

	Age	Male	Year	Deaths	Population
<b>1</b>	35	0	1999	1291	1578829
<b>2</b>	35	0	2000	1264	1528463
<b>3</b>	35	0	2001	1186	1377466
<b>4</b>	35	0	2002	1194	1333639
<b>5</b>	35	0	2003	1166	1302188
<b>6</b>	35	0	2004	1166	1325435
<b>7</b>	35	0	2005	1201	1383427
<b>8</b>	35	0	2006	1197	1349975
<b>9</b>	35	0	2007	1147	1239648
<b>10</b>	35	0	2008	1098	1155005
⋮ more					
<b>900</b>	64	1	2013	17761	1255101

```
• white_nonhisp_by_sex =  
  CSV.read(joinpath(expanduser("~"),  
    ".julia", "dev",  
    "RegressionAndOtherStories", "data",  
    "AgePeriodCohort",  
    "white_nonhisp_death_rates_by_sex.txt"),  
    DataFrame; delim="\t")
```