STAT 5428: Homework 1

Solayman Hossain Emon

Student ID: 80744292

In [1]:
```
using CSV, DataFrames, Statistics, StatsBase, Gadfly, StatsPlots, PrettyTables, Plots
```

In [2]:
```
diabetes= CSV.read("diabetes.csv", DataFrame);
```

In [3]:
```
sz = size(diabetes)
```

Out[3]:
```
(768, 9)
```

In [4]:
```
num_ftr = sz[2]
```

Out[4]:
```
9
```

In [5]:
```
describe(diabetes)
```

Out[5]: 9 rows × 7 columns

	variable	mean	min	median	max	nmissing	eltype
	Symbol	Float64	Real	Float64	Real	Int64	DataType
1	Pregnancies	3.84505	0	3.0	17	0	Int64
2	Glucose	120.895	0	117.0	199	0	Int64
3	BloodPressure	69.1055	0	72.0	122	0	Int64
4	SkinThickness	20.5365	0	23.0	99	0	Int64
5	Insulin	79.7995	0	30.5	846	0	Int64

	variable	mean	min	median	max	nmissing	eltype
	Symbol	Float64	Real	Float64	Real	Int64	DataType
6	BMI	31.9926	0.0	32.0	67.1	0	Float64
7	DiabetesPedigreeFunction	0.471876	0.078	0.3725	2.42	0	Float64
8	Age	33.2409	21	29.0	81	0	Int64
9	Outcome	0.348958	0	0.0	1	0	Int64

In [6]:

```
fnames = names(diabetes)
```

Out[6]:

```
9-element Vector{String}:
 "Pregnancies"
 "Glucose"
 "BloodPressure"
 "SkinThickness"
 "Insulin"
 "BMI"
 "DiabetesPedigreeFunction"
 "Age"
 "Outcome"
```

Interquartile Range (IQR)

- ### IQR = Q3(Third Quartile Value) - Q1(First Quartile Value)

In [7]:

```
function iqr(samples)
    samples = sort(samples)

    # Get the size of the samples
    samples_len = length(samples)

    # Divide the size by 2
    sub_samples_len = div(samples_len, 2)

    # Know the indexes
    start_index_of_q1 = 1
    end_index_of_q1 = sub_samples_len
    start_index_of_q3 = samples_len - sub_samples_len + 1
    end_index_of_q3 = samples_len

    # Q1 median value
    median_value_of_q1 = median(view(samples, start_index_of_q1:end_index_of_q1))

    # Q2 median value
    median_value_of_q3 = median(view(samples, start_index_of_q3:end_index_of_q3))

    # Find the IQR value
    iqr_result = median_value_of_q3 - median_value_of_q1
    return iqr_result
end
```

Out[7]:
```
iqr (generic function with 1 method)
```

Calculating Mean, Median, Standard Deviation, Skewness, Kurtosis, IQR

In [8]:
```
header = (["Variable", "Mean", "Median", "Standard Deviation", "Skewness", "Kurtosis", "IQR"])
mn = Vector{Float64}()
mdn = Vector{Float64}()
Std = Vector{Float64}()
Skw = Vector{Float64}()
Krt = Vector{Float64}()
IQR = Vector{Float64}()

for vname in fnames[1:num_ftr]
#println(vname, ":", mean(diabetes[!, Symbol(vname)]), median(diabetes[!, Symbol(vname)]), std(diabetes[!, Symbol(vname)]
append!(mn, [ mean(diabetes[!, Symbol(vname)])])
append!(mdn, [ median(diabetes[!, Symbol(vname)])])
append!(Std, [ std(diabetes[!, Symbol(vname)])])
append!(Skw, [ skewness(diabetes[!, Symbol(vname)])])
append!(Krt, [ kurtosis(diabetes[!, Symbol(vname)])])
append!(IQR, [ iqr(diabetes[!, Symbol(vname)])])
end


Data = DataFrame(A=fnames, B=mn, C = mdn, D=Std, E=Skw, F=Krt, G=IQR)
pretty_table(Data, header=header, tf = tf_unicode_rounded)
```

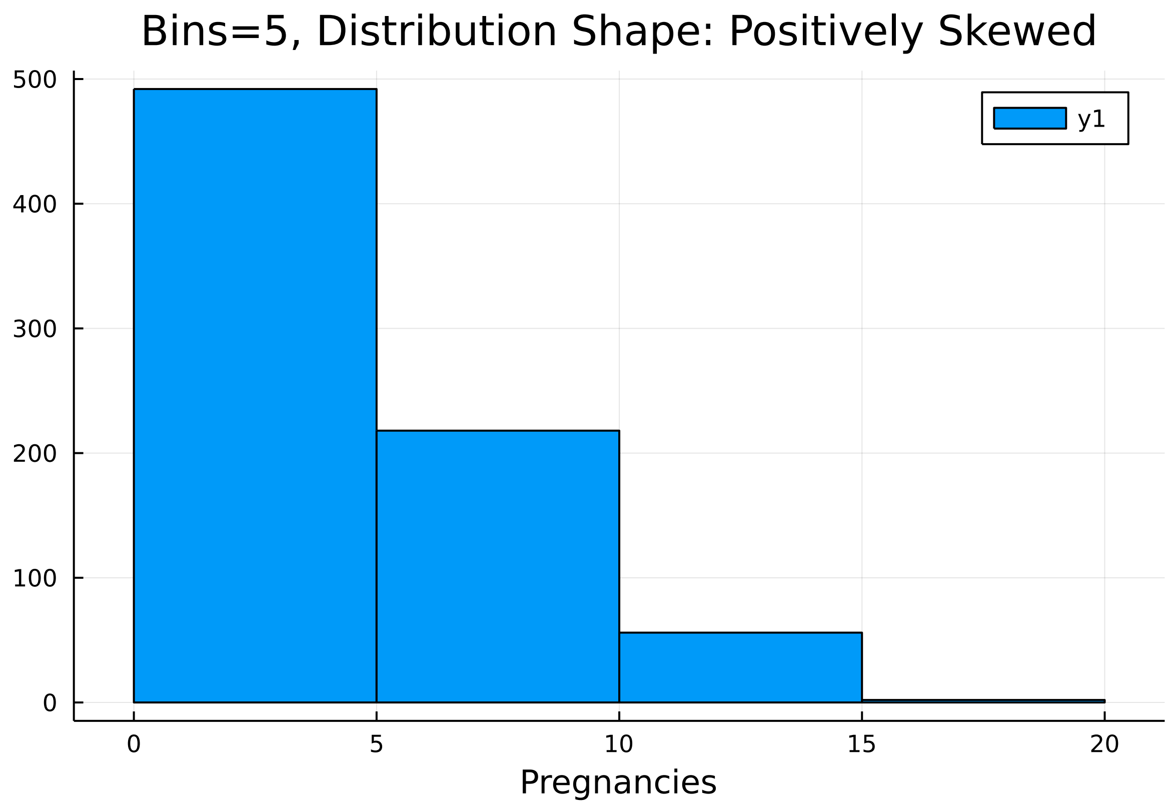
```
│                 Variable │     Mean │ Median │ Standard Deviation │  Skewness │  Kurtosis │   IQR │
├──────────────────────────┼──────────┼────────┼────────────────────┼───────────┼───────────┼───────┤
│              Pregnancies │  3.84505 │    3.0 │            3.36958 │  0.899912 │  0.150383 │   5.0 │
│                  Glucose │  120.895 │  117.0 │            31.9726 │  0.173414 │  0.628813 │  41.5 │
│            BloodPressure │  69.1055 │   72.0 │            19.3558 │  -1.84001 │   5.13869 │  18.0 │
│            SkinThickness │  20.5365 │   23.0 │            15.9522 │  0.109159 │ -0.524494 │  32.0 │
│                  Insulin │  79.7995 │   30.5 │            115.244 │   2.26781 │   7.15957 │ 127.5 │
│                      BMI │  31.9926 │   32.0 │            7.88416 │ -0.428143 │   3.26126 │   9.3 │
│ DiabetesPedigreeFunction │ 0.471876 │ 0.3725 │           0.331329 │   1.91616 │   5.55079 │ 0.383 │
│                      Age │  33.2409 │   29.0 │            11.7602 │   1.12739 │  0.631177 │  17.0 │
│                  Outcome │ 0.348958 │    0.0 │           0.476951 │  0.633776 │  -1.59833 │   1.0 │
└──────────────────────────┴──────────┴────────┴────────────────────┴───────────┴───────────┴───────┘
```

Histogram for all Variables

In [9]:

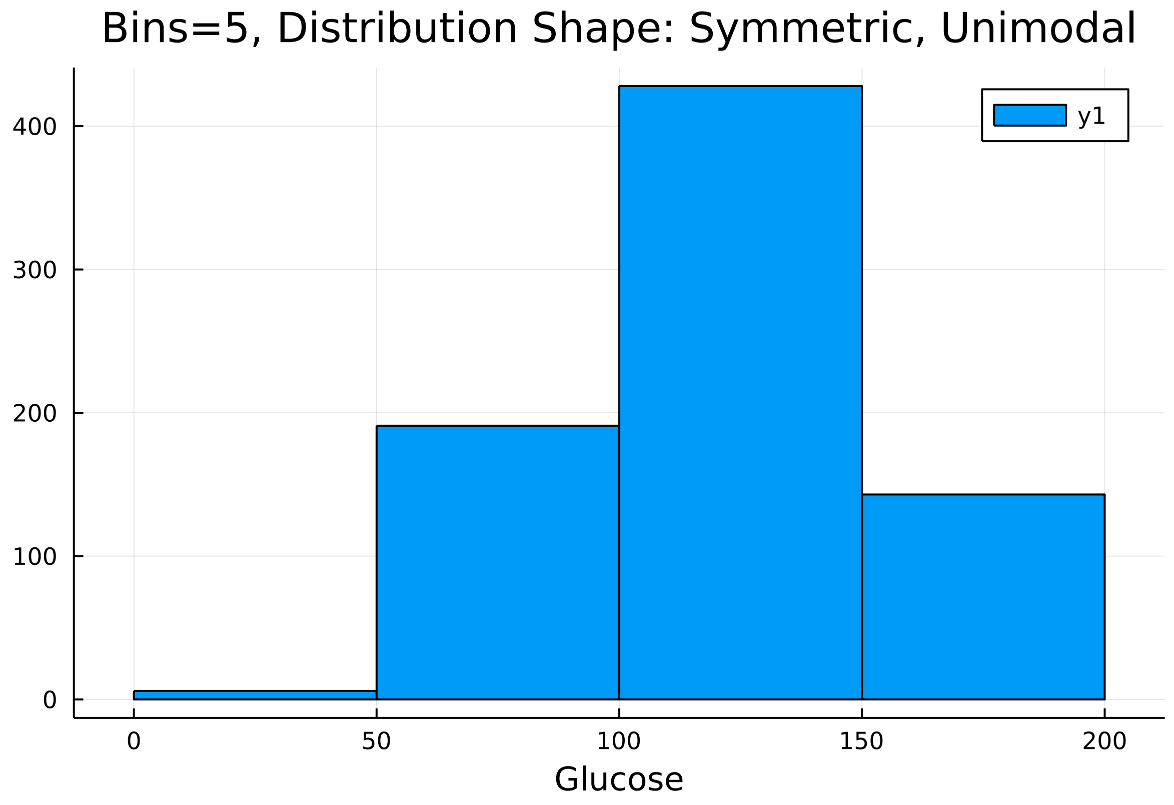
```julia
h1 = Plots.histogram(diabetes[!, :Pregnancies], bins = 5, xlabel = "Pregnancies", title = "Bins=5, Distribution Shape: Po
```

Out[9]:



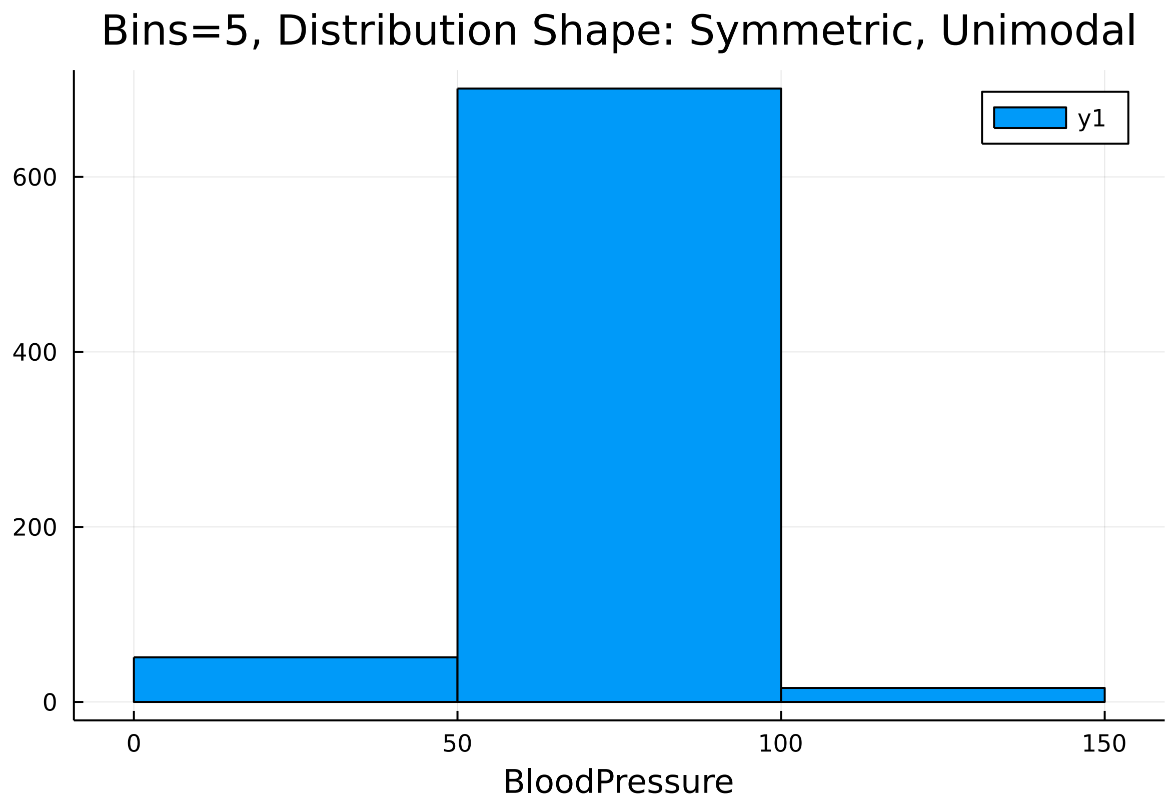
In [10]:
```
h2 = Plots.histogram(diabetes[!, :Glucose], bins = 5, xlabel = "Glucose",  title = "Bins=5, Distribution Shape: Symmetric
```

Out[10]:



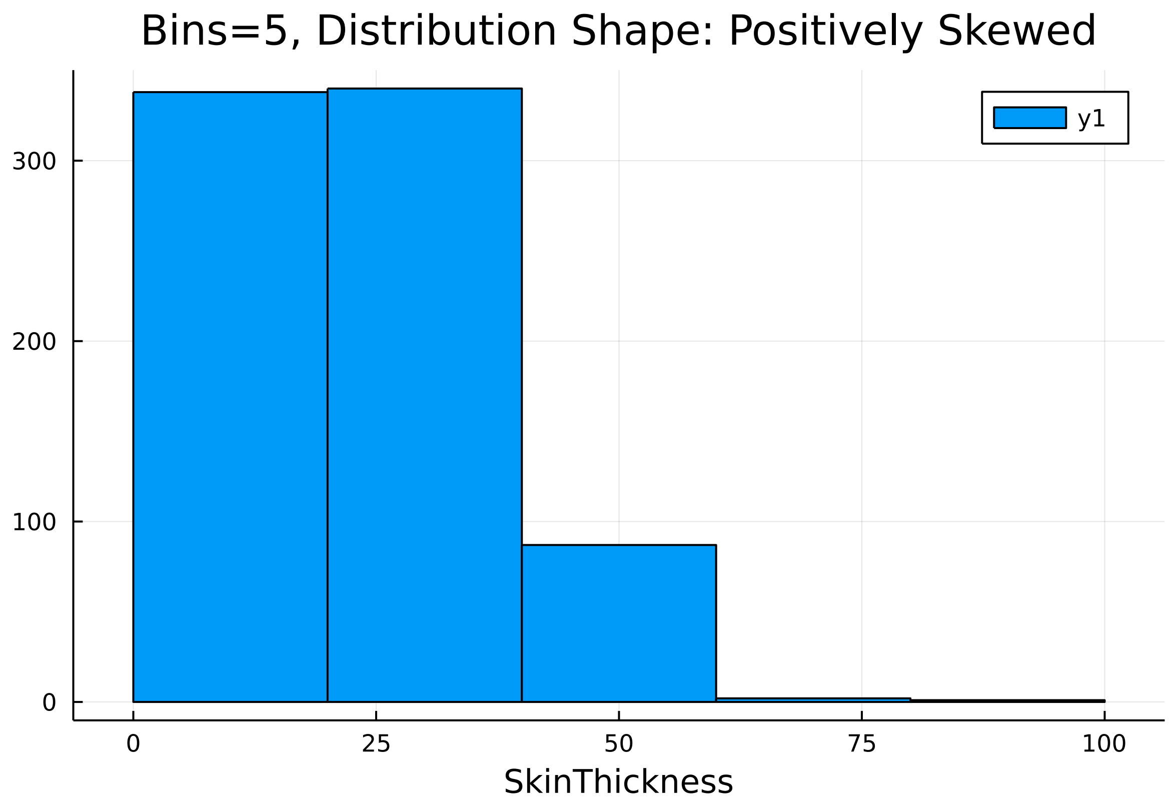
In [11]:
```
h3 = Plots.histogram(diabetes[!, :BloodPressure], bins = 5, xlabel = "BloodPressure",  title = "Bins=5, Distribution Shap
```

Out[11]:



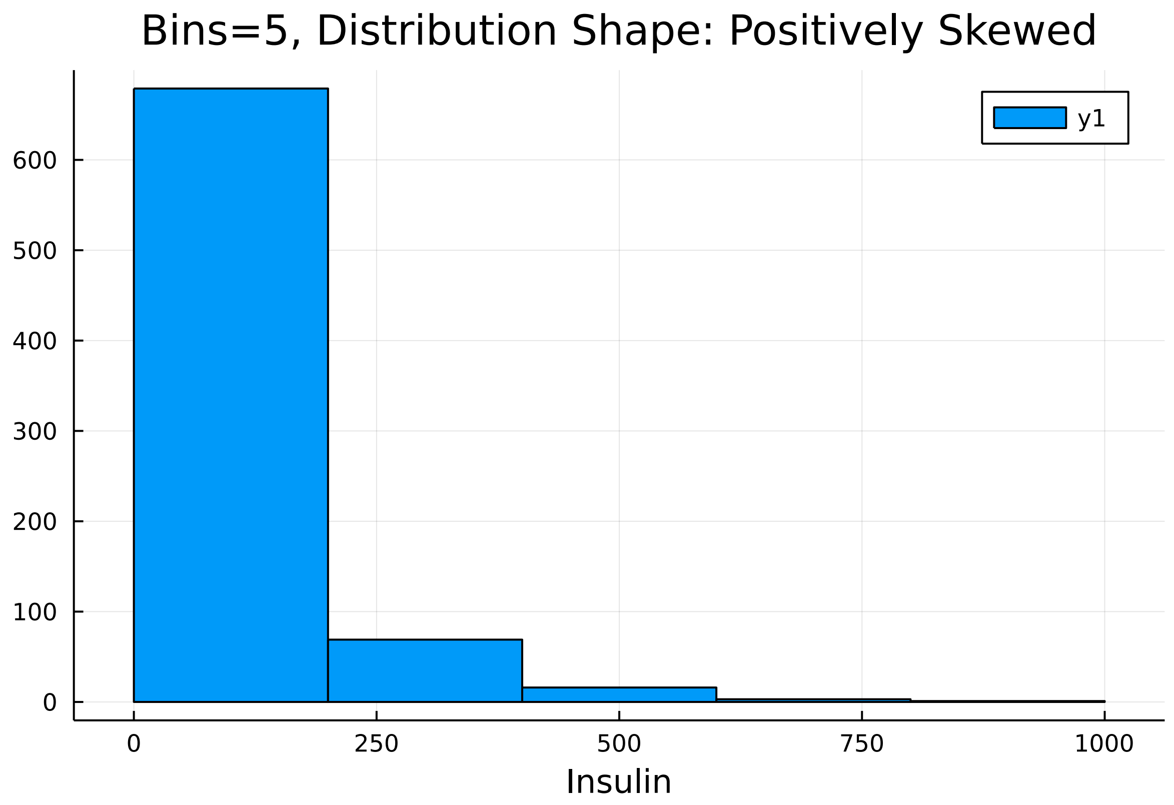
In [12]:
```
h4 = Plots.histogram(diabetes[!, :SkinThickness], bins = 5, xlabel = "SkinThickness",  title = "Bins=5, Distribution Shap
```

Out[12]:



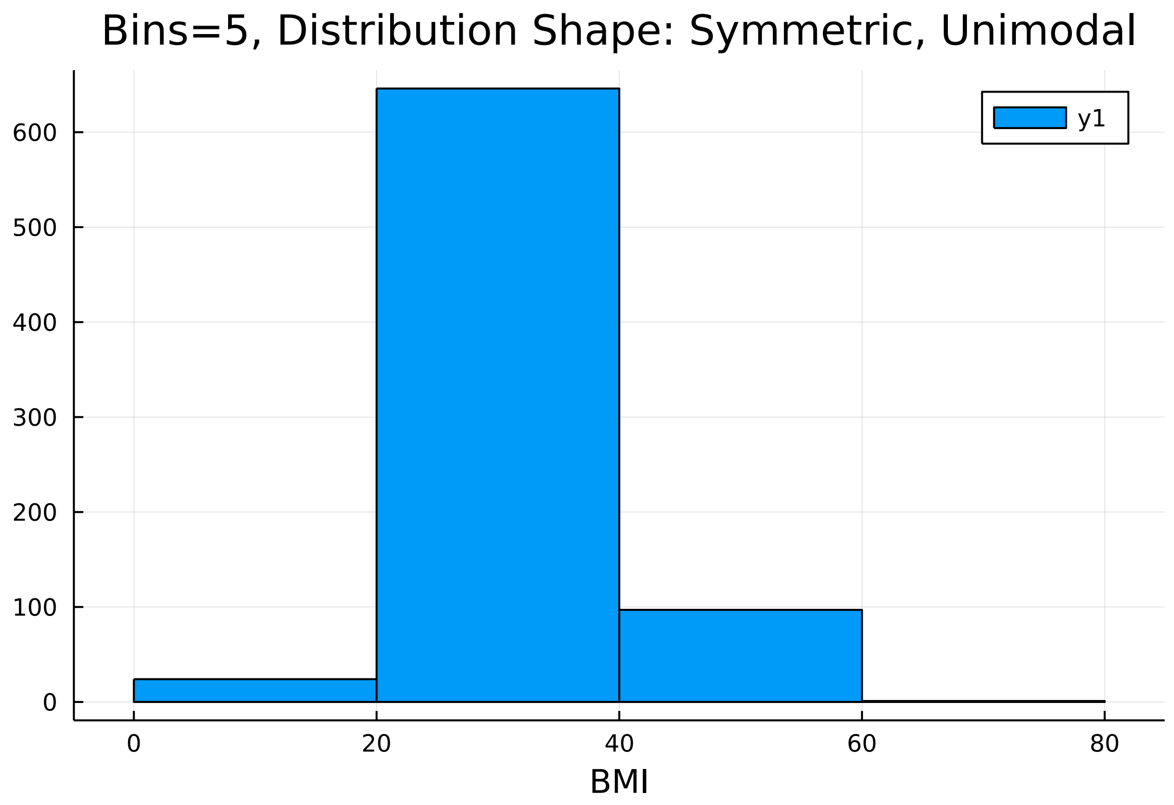
```
In [13]: h5 = Plots.histogram(diabetes[!, :Insulin], bins = 5, xlabel = "Insulin",  title = "Bins=5, Distribution Shape: Positivel
```

```
Out[13]:
```



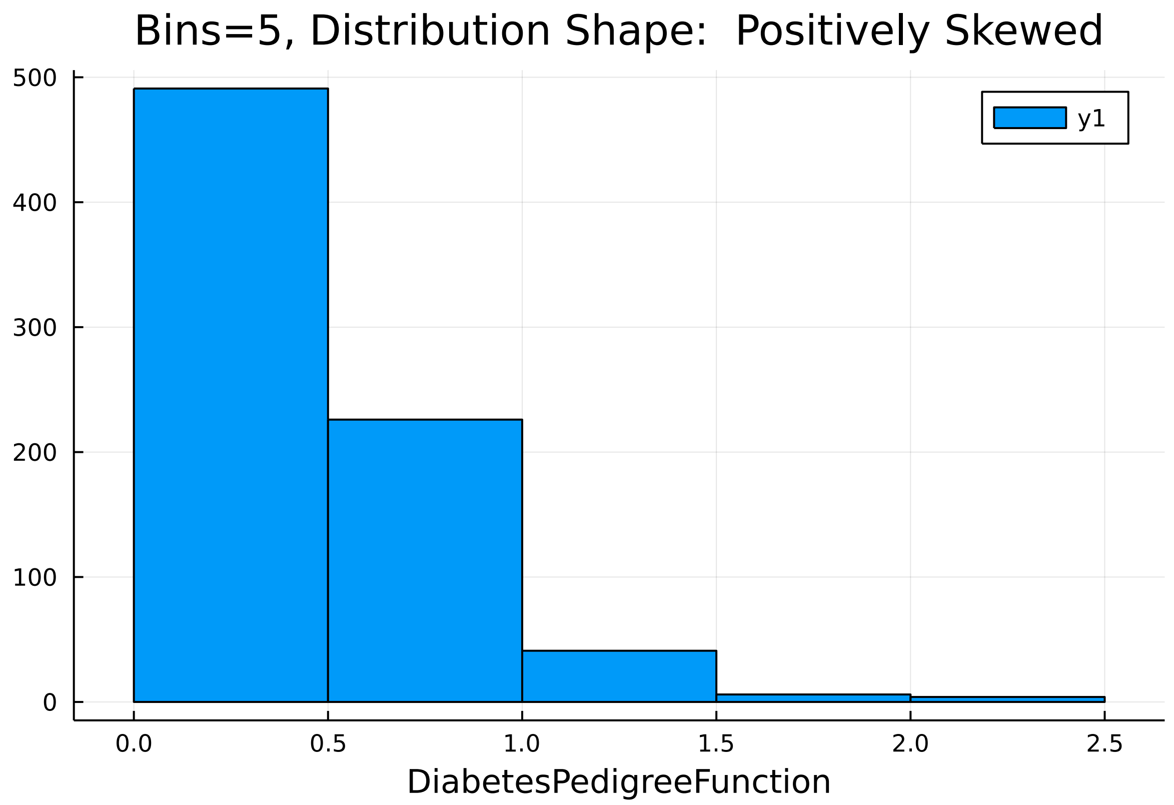
In [14]:
```
h6 = Plots.histogram(diabetes[!, :BMI], bins = 5, xlabel = "BMI",  title = "Bins=5, Distribution Shape: Symmetric, Unimoc
```

Out[14]:



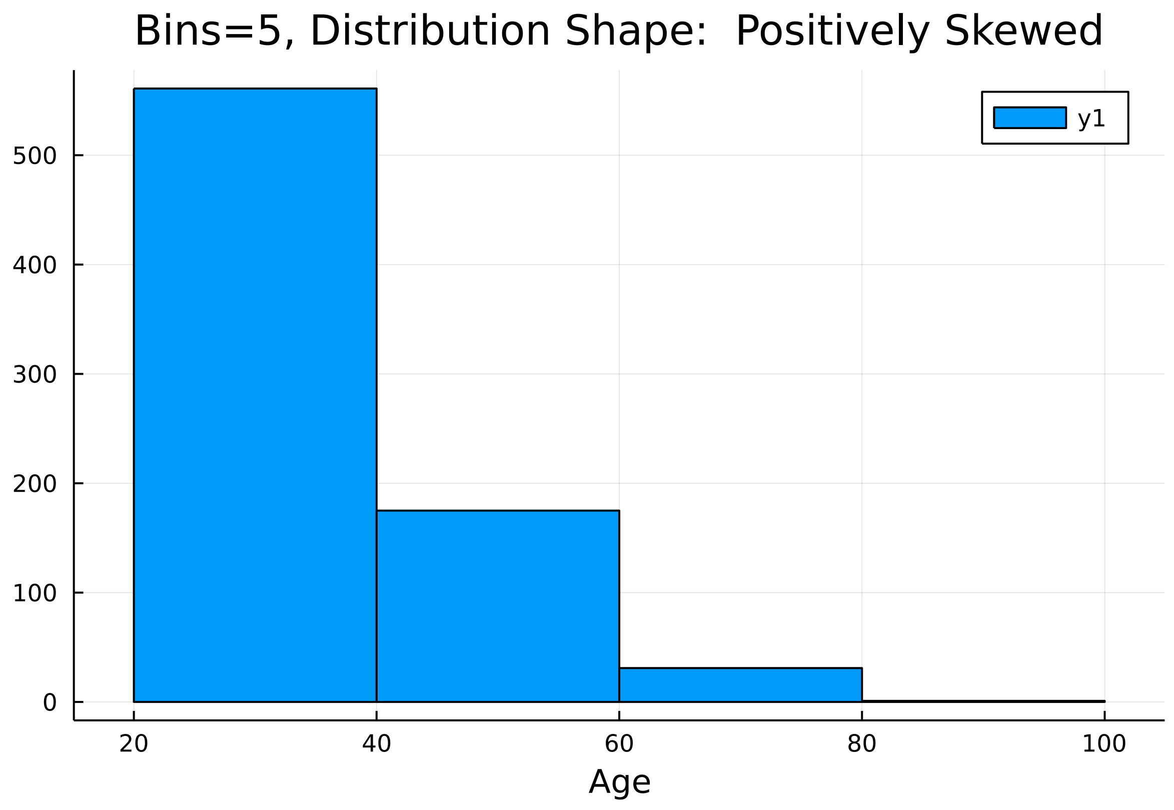
```
In [15]: h7 = Plots.histogram(diabetes[!, :DiabetesPedigreeFunction], bins = 5, xlabel = "DiabetesPedigreeFunction", title = "Bins
```

```
Out[15]:
```



In [16]:
```
h8 = Plots.histogram(diabetes[!, :Age], bins = 5, xlabel = "Age", title = "Bins=5, Distribution Shape:  Positively Skewec
```

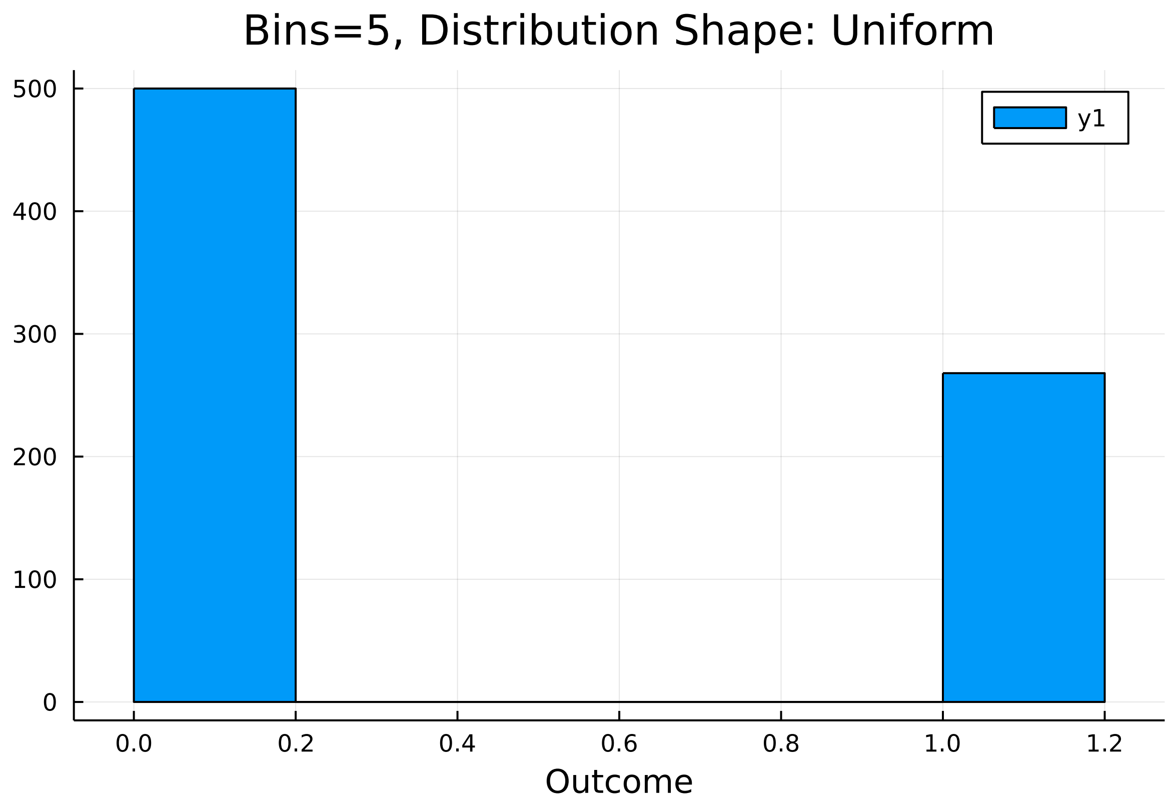
Out[16]:



Distribution Shape: Positively Skewed

In [17]:
```
h9 = Plots.histogram(diabetes[!, :Outcome], bins = 5, xlabel = "Outcome", title = "Bins=5, Distribution Shape: Uniform")
```

Out[17]:



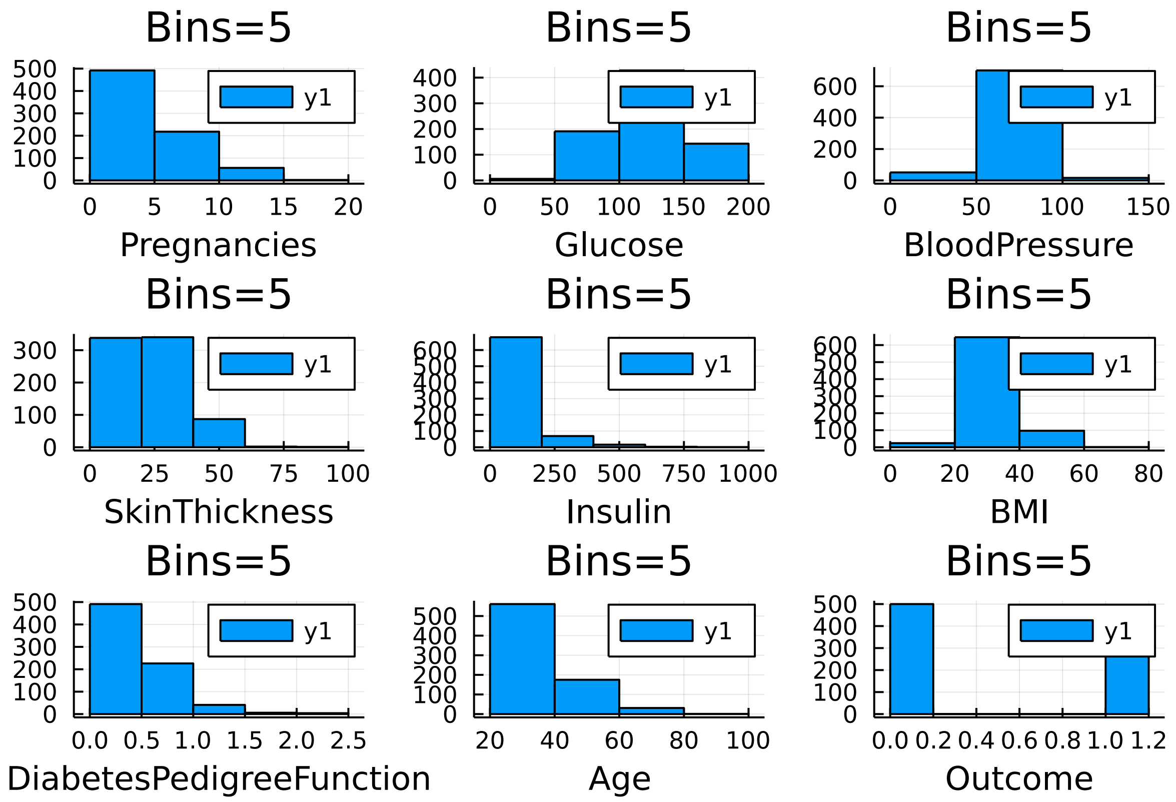
Stack Histograms Together to Compare

In [18]:

```
h11 = Plots.histogram(diabetes[!, :Pregnancies], bins = 5, xlabel = "Pregnancies", title = "Bins=5");
h22 = Plots.histogram(diabetes[!, :Glucose], bins = 5, xlabel = "Glucose", title = "Bins=5");
h33 = Plots.histogram(diabetes[!, :BloodPressure], bins = 5, xlabel = "BloodPressure", title = "Bins=5");
h44 = Plots.histogram(diabetes[!, :SkinThickness], bins = 5, xlabel = "SkinThickness", title = "Bins=5");
h55 = Plots.histogram(diabetes[!, :Insulin], bins = 5, xlabel = "Insulin", title = "Bins=5");
h66 = Plots.histogram(diabetes[!, :BMI], bins = 5, xlabel = "BMI", title = "Bins=5");
h77 = Plots.histogram(diabetes[!, :DiabetesPedigreeFunction], bins = 5, xlabel = "DiabetesPedigreeFunction", title = "Bir
h88 = Plots.histogram(diabetes[!, :Age], bins = 5, xlabel = "Age", title = "Bins=5");
h99 = Plots.histogram(diabetes[!, :Outcome], bins = 5, xlabel = "Outcome", title = "Bins=5");

plot!(h11, h22, h33, h44, h55, h66, h77, h88, h99)
```

Out[18]:



In term of shape:

- ### Pregnancies, Insulin, Age, DiabetesPedigreeFunction, SkinThickness are close to similar
- ### Glucose, BloodPressure, BMI are nearly similar type
- ### Outcome type is totally different as it is a categorical variable

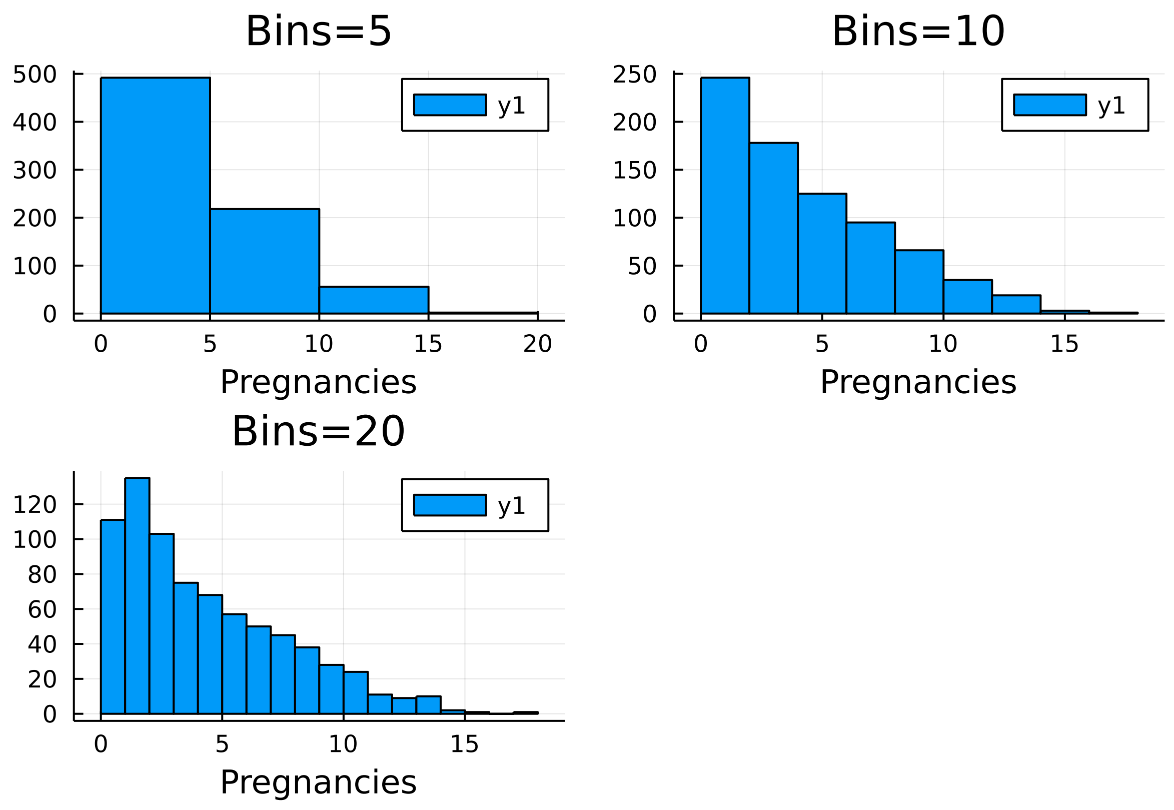
Different Bin Size

- ### Bin Width = (max value of data – min value of data) / total number of bins

- ### According to the number of bins, the shape of the histograms will change. In Julia, if 'bins' variable is varied the shape of the histogram will be also changed. In our experiment, we set 'Bin Width' as 5, 10, 20 respectively.

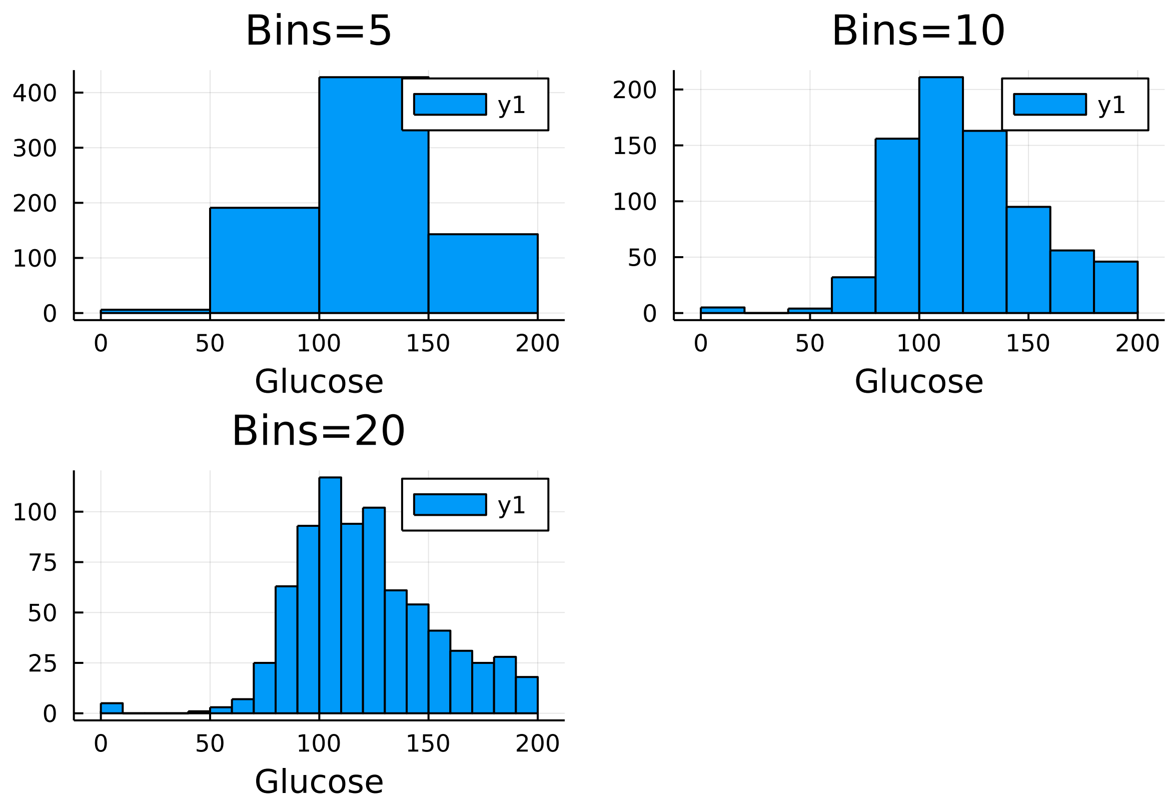
In [19]:
```
h1b = Plots.histogram(diabetes[!, :Pregnancies], bins = 10, xlabel = "Pregnancies", title = "Bins=10");
h1bb = Plots.histogram(diabetes[!, :Pregnancies], bins = 20, xlabel = "Pregnancies", title = "Bins=20");
plot!(h11, h1b, h1bb)
```

Out[19]:



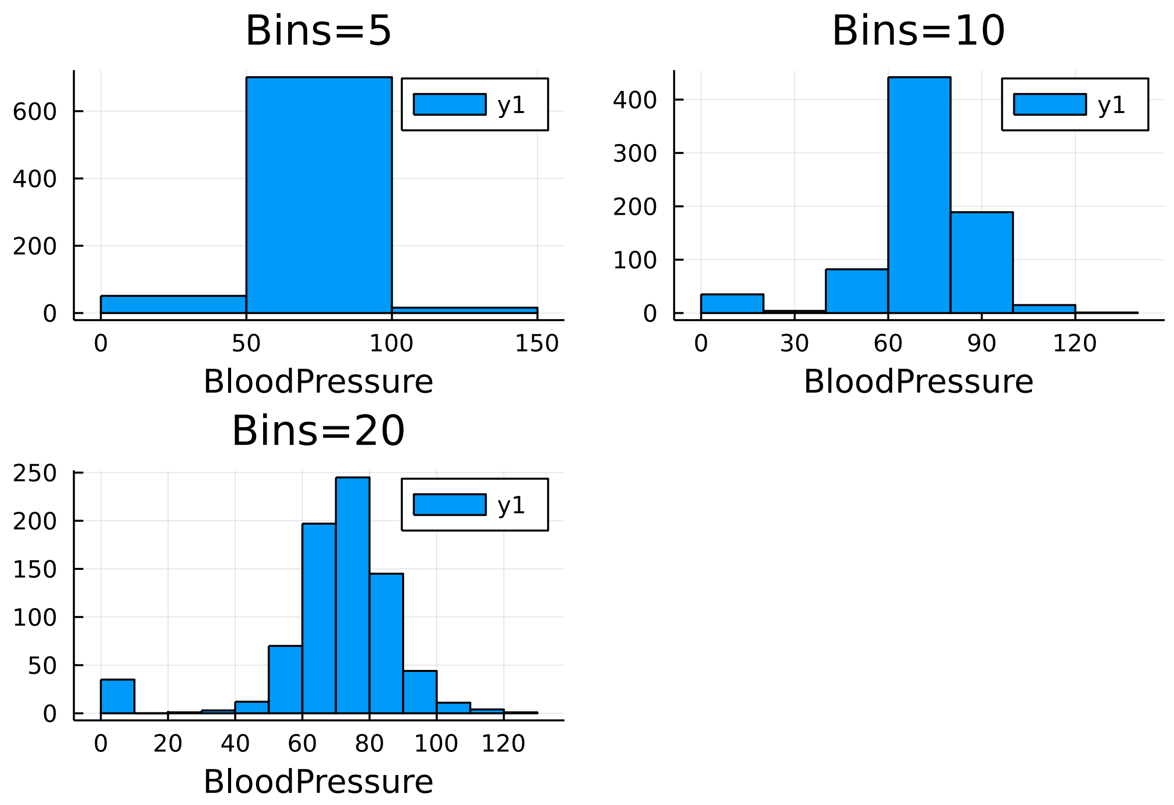
In [20]:
```
h2b = Plots.histogram(diabetes[!, :Glucose], bins = 10, xlabel = "Glucose", title = "Bins=10");
h2bb = Plots.histogram(diabetes[!, :Glucose], bins = 20, xlabel = "Glucose", title = "Bins=20");
plot!(h22, h2b, h2bb)
```

Out[20]:



In [21]:
```
h3b = Plots.histogram(diabetes[!, :BloodPressure], bins = 10, xlabel = "BloodPressure", title = "Bins=10");
h3bb = Plots.histogram(diabetes[!, :BloodPressure], bins = 20, xlabel = "BloodPressure", title = "Bins=20");
plot!(h33, h3b, h3bb)
```

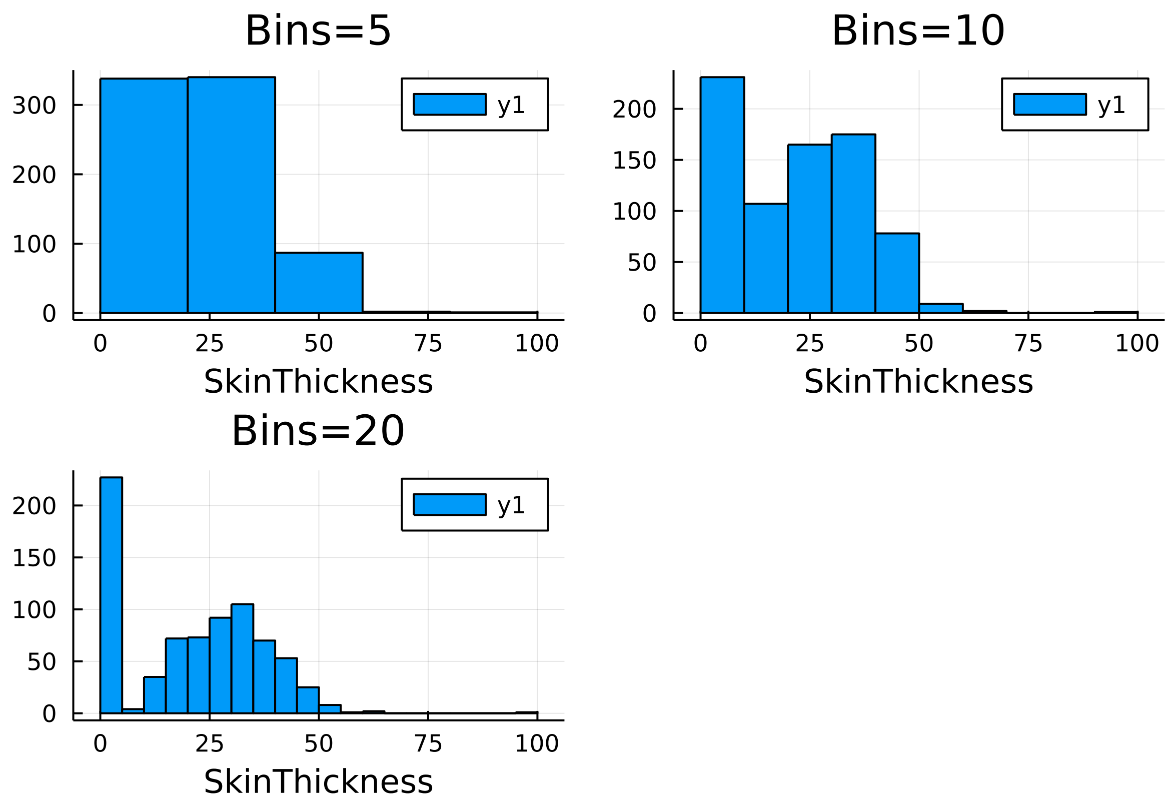
Out[21]:



In [22]:

```
h4b = Plots.histogram(diabetes[!, :SkinThickness], bins = 10, xlabel = "SkinThickness", title = "Bins=10");
h4bb = Plots.histogram(diabetes[!, :SkinThickness], bins = 20, xlabel = "SkinThickness", title = "Bins=20");
plot!(h44, h4b, h4bb)
```

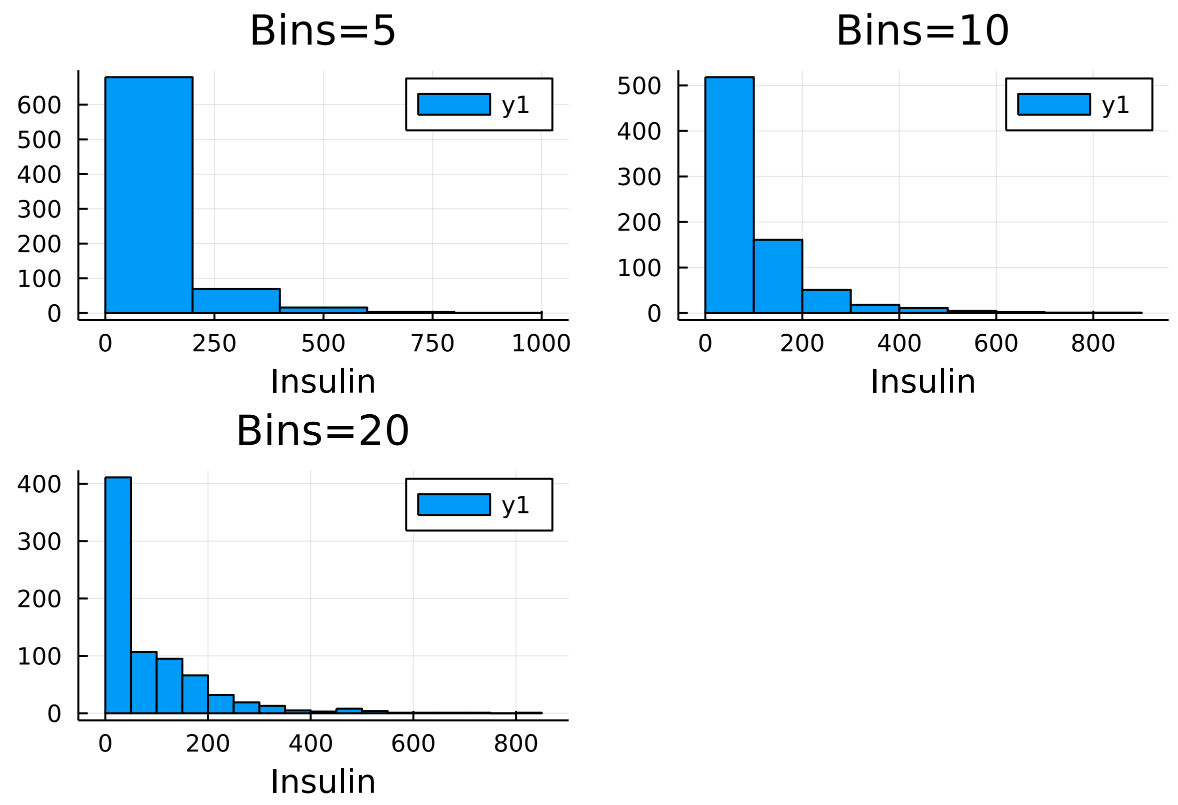
Out[22]:



In [23]:

```
h5b = Plots.histogram(diabetes[!, :Insulin], bins = 10, xlabel = "Insulin", title = "Bins=10");
h5bb = Plots.histogram(diabetes[!, :Insulin], bins = 20, xlabel = "Insulin", title = "Bins=20");
plot!(h55, h5b, h5bb)
```

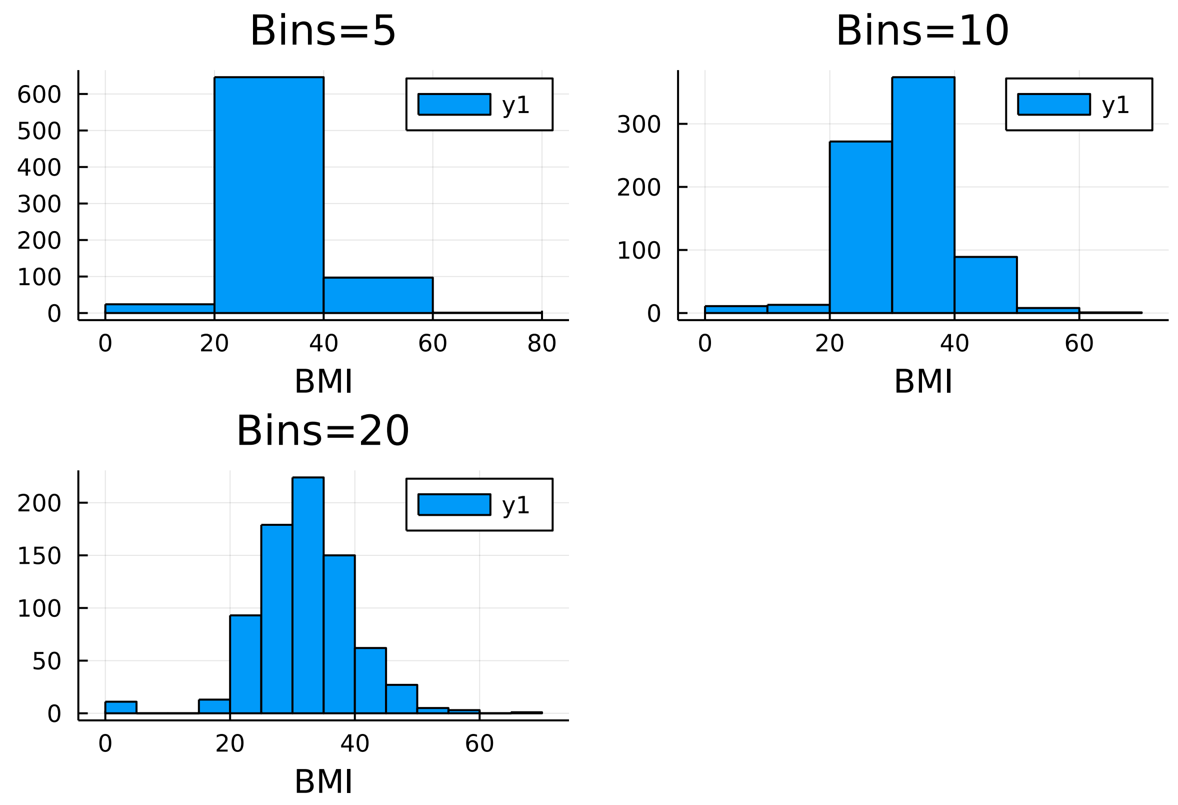
Out[23]:



In [24]:

```
h6b = Plots.histogram(diabetes[!, :BMI], bins = 10, xlabel = "BMI", title = "Bins=10");
h6bb = Plots.histogram(diabetes[!, :BMI], bins = 20, xlabel = "BMI", title = "Bins=20");
plot!(h66, h6b, h6bb)
```

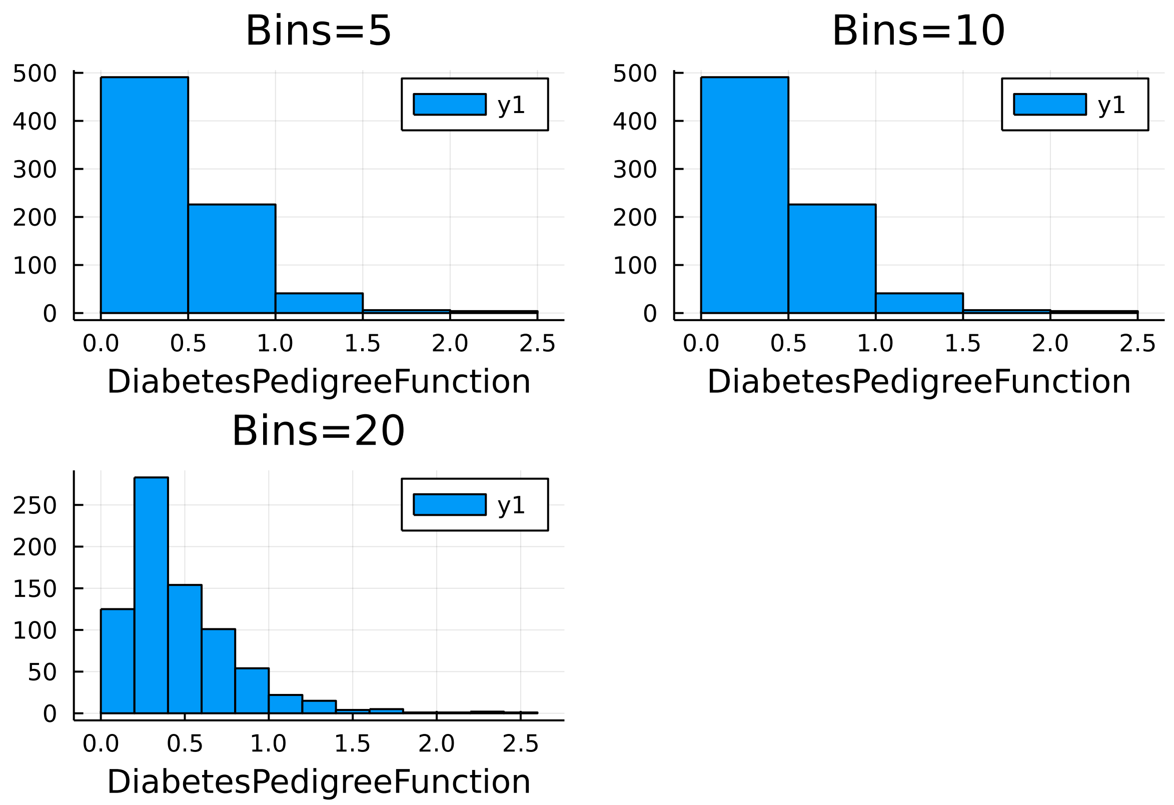
Out[24]:



In [25]:

```
h7b = Plots.histogram(diabetes[!, :DiabetesPedigreeFunction], bins = 10, xlabel = "DiabetesPedigreeFunction", title = "Bi
h7bb = Plots.histogram(diabetes[!, :DiabetesPedigreeFunction], bins = 20, xlabel = "DiabetesPedigreeFunction", title = "E
plot!(h77, h7b, h7bb)
```

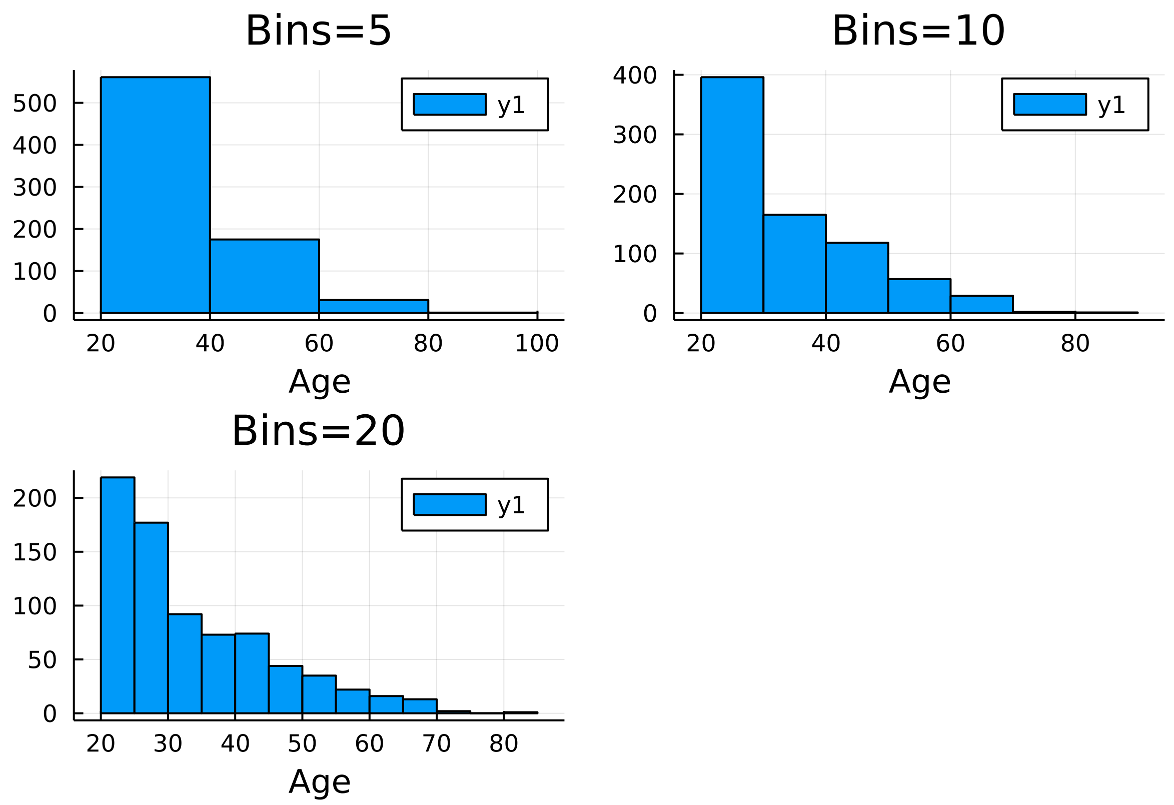
Out[25]:



In [26]:

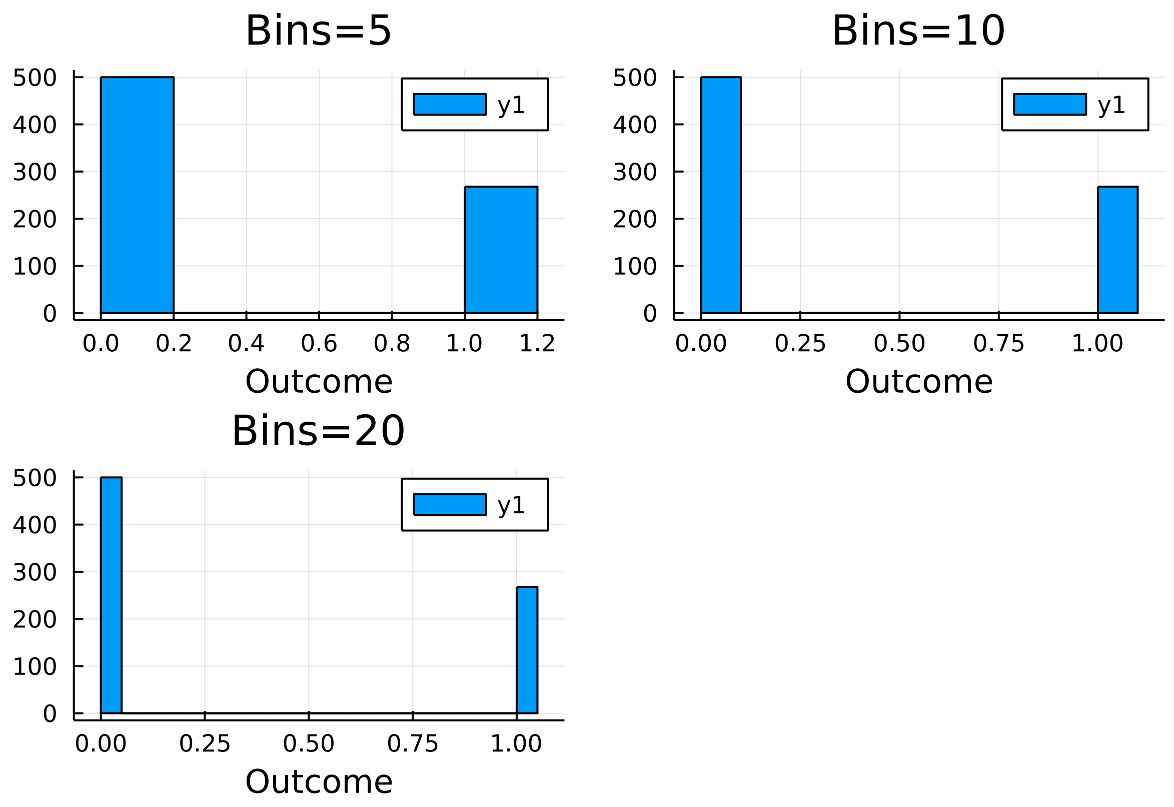
```
h8b = Plots.histogram(diabetes[!, :Age], bins = 10, xlabel = "Age", title = "Bins=10");
h8bb = Plots.histogram(diabetes[!, :Age], bins = 20, xlabel = "Age", title = "Bins=20");
plot!(h88, h8b, h8bb)
```

Out[26]:



In [27]:
```
h9b = Plots.histogram(diabetes[!, :Outcome], bins = 10, xlabel = "Outcome", title = "Bins=10");
h9bb = Plots.histogram(diabetes[!, :Outcome], bins = 20, xlabel = "Outcome", title = "Bins=20");
plot!(h99, h9b, h9bb)
```

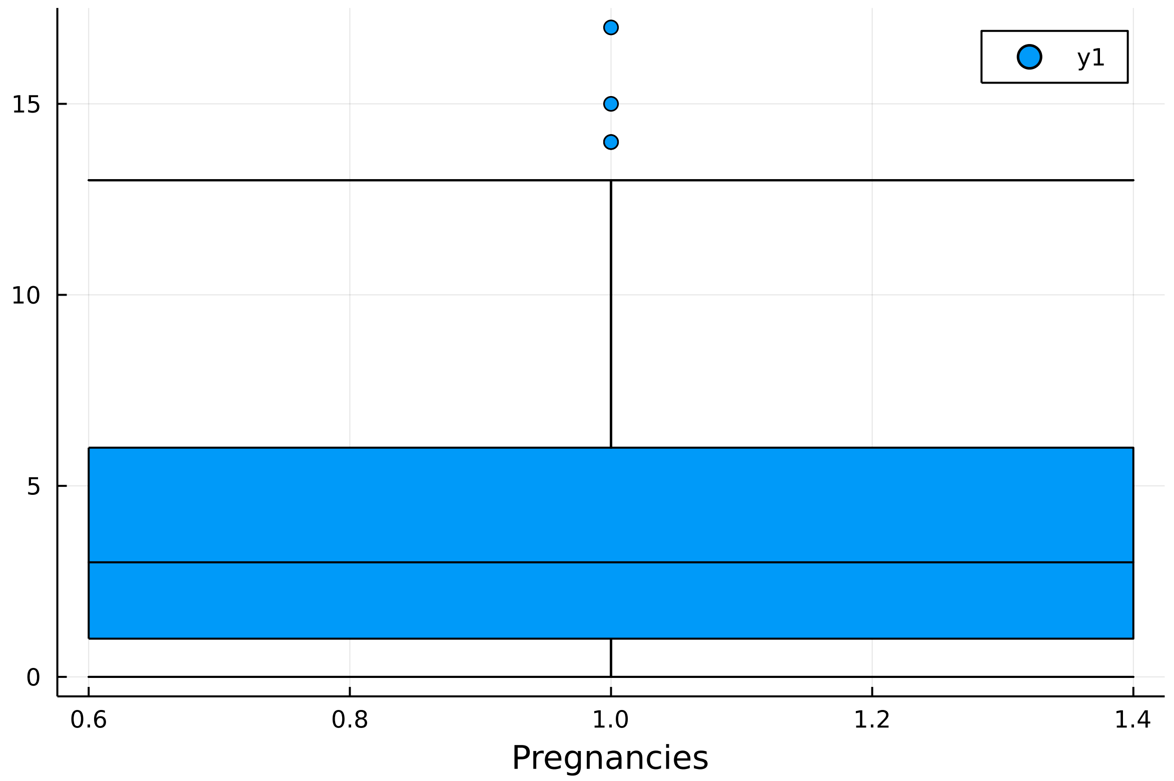
Out[27]:



Box-and-Whisker Plots for the All Variables

In [28]:
```
b1 = StatsPlots.boxplot(diabetes[!, :Pregnancies], xlabel = "Pregnancies")
```

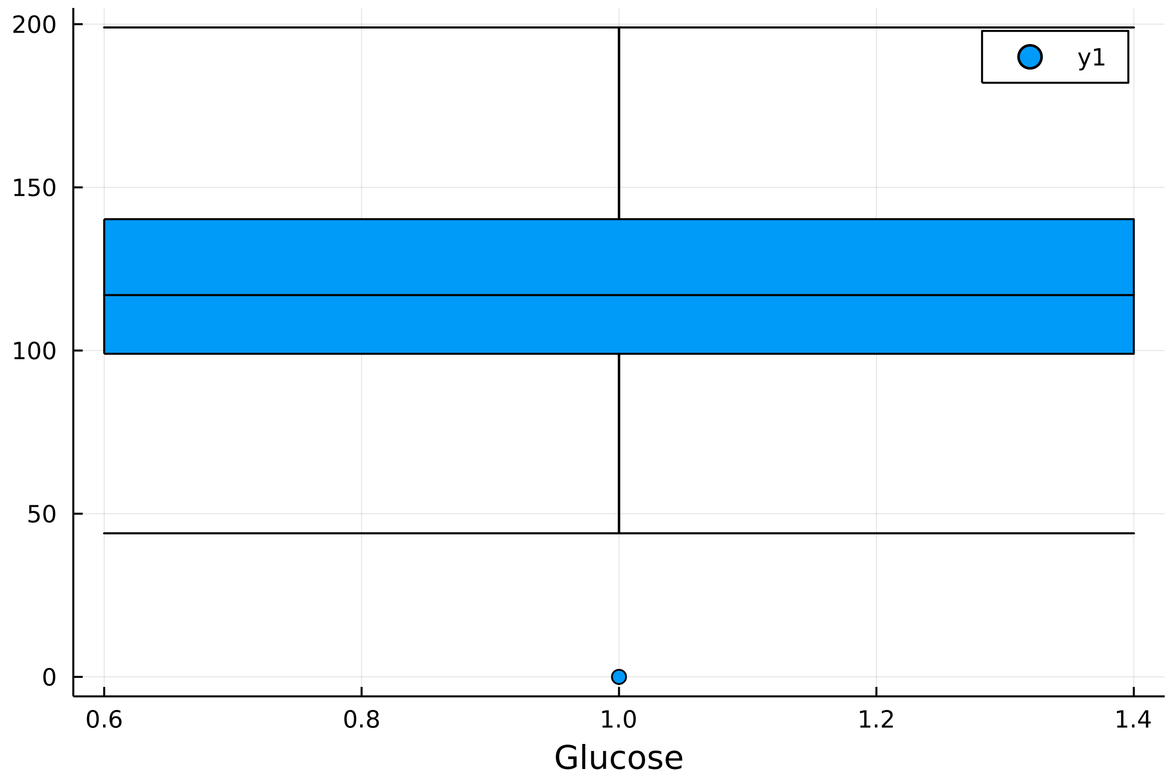
Out[28]:



In [29]:

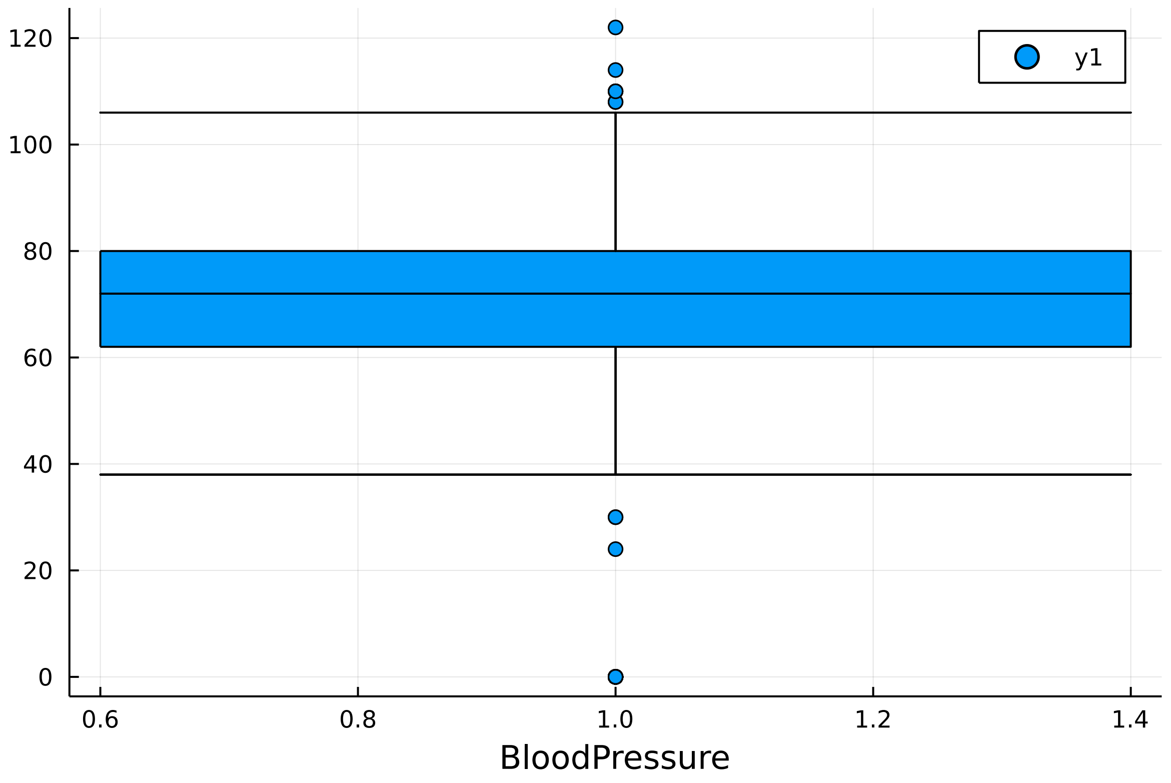
```
b2 = StatsPlots.boxplot(diabetes[!, :Glucose], xlabel = "Glucose")
```

Out[29]:



In [30]:
```
b3 = StatsPlots.boxplot(diabetes[!, :BloodPressure], xlabel = "BloodPressure")
```

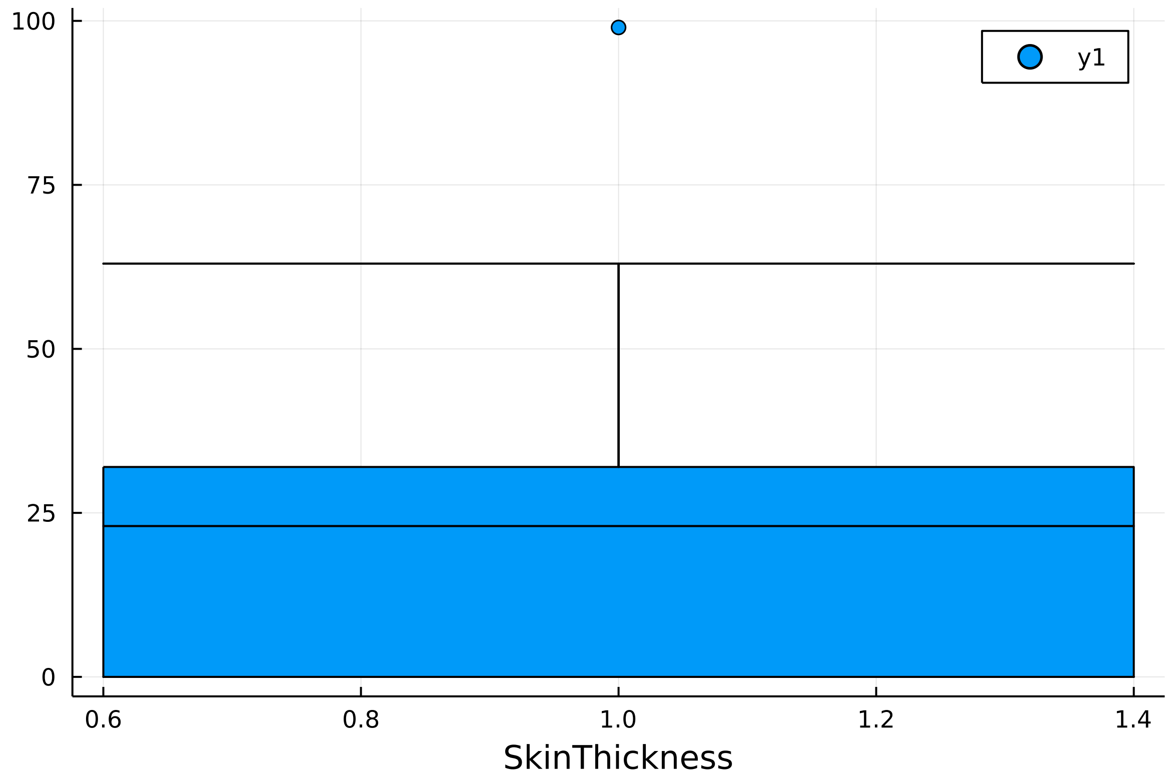
Out[30]:



In [31]:

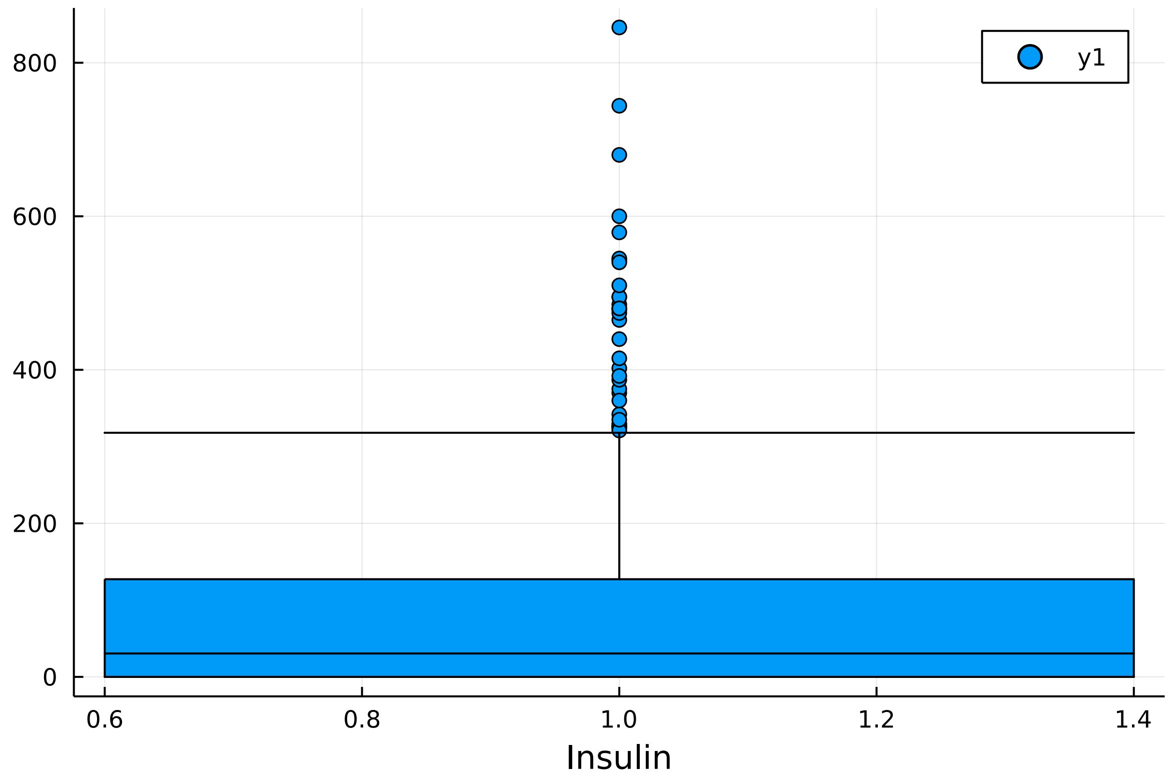
```
b4 = StatsPlots.boxplot(diabetes[!, :SkinThickness], xlabel = "SkinThickness")
```

Out[31]:



In [32]:
```
b5 = StatsPlots.boxplot(diabetes[!, :Insulin], xlabel = "Insulin")
```

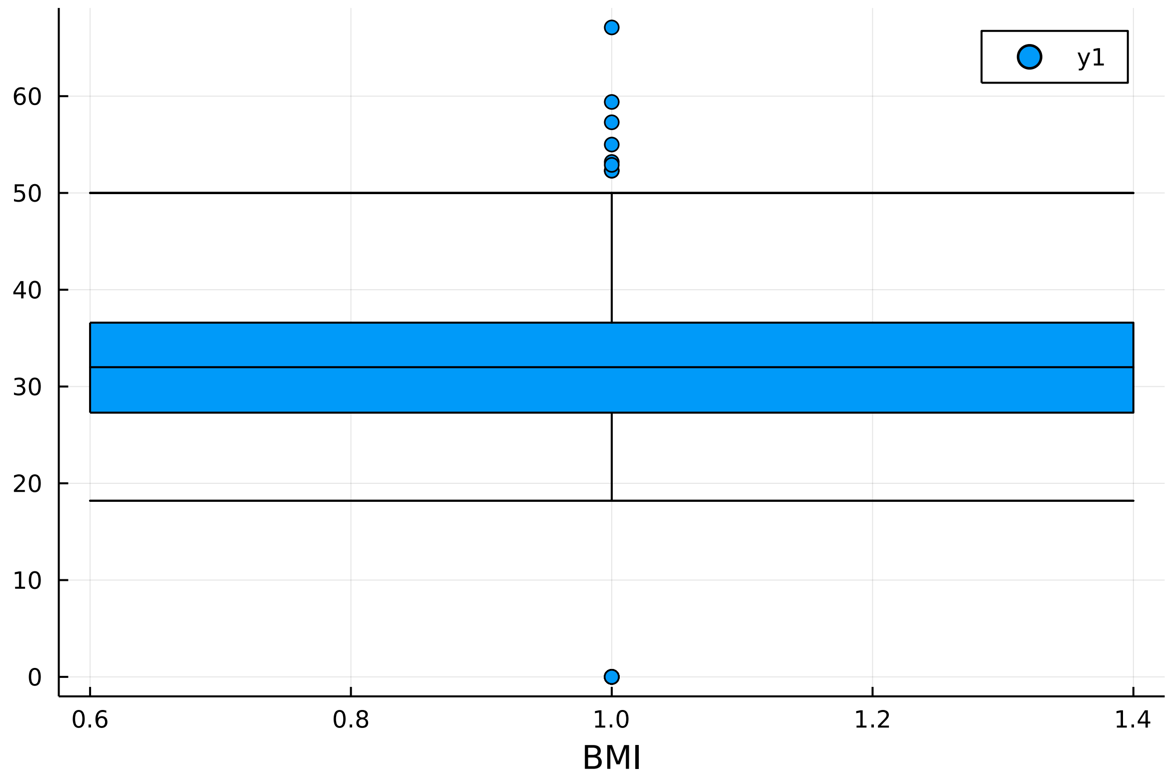
Out[32]:



In [33]:

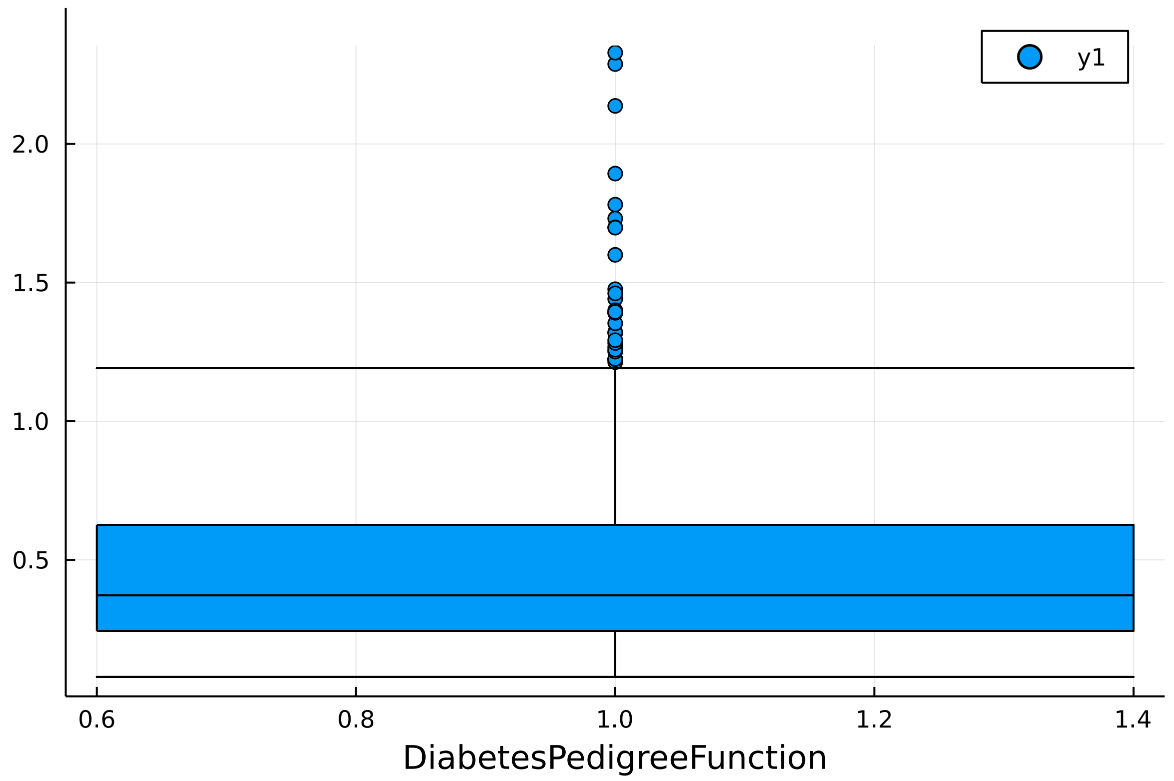
```
b6 = StatsPlots.boxplot(diabetes[!, :BMI], xlabel = "BMI")
```

Out[33]:



In [34]:
```
b7 = StatsPlots.boxplot(diabetes[!, :DiabetesPedigreeFunction], xlabel = "DiabetesPedigreeFunction")
```

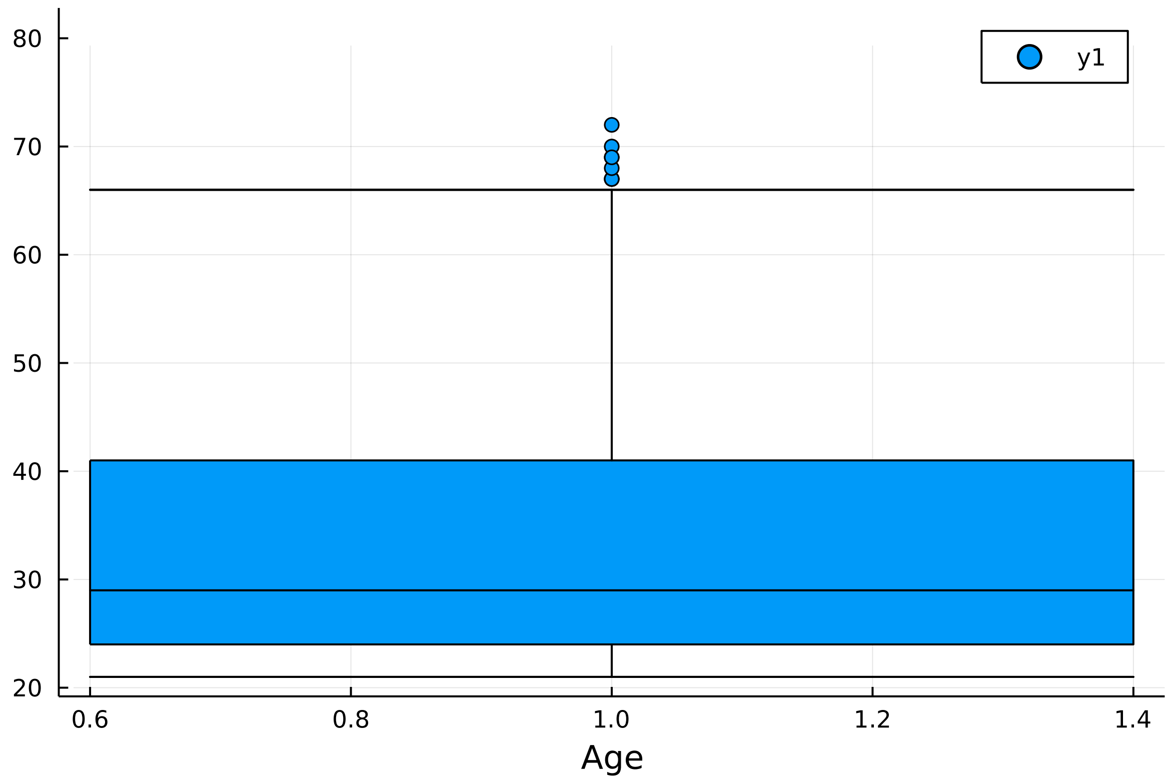
Out[34]:



In [35]:

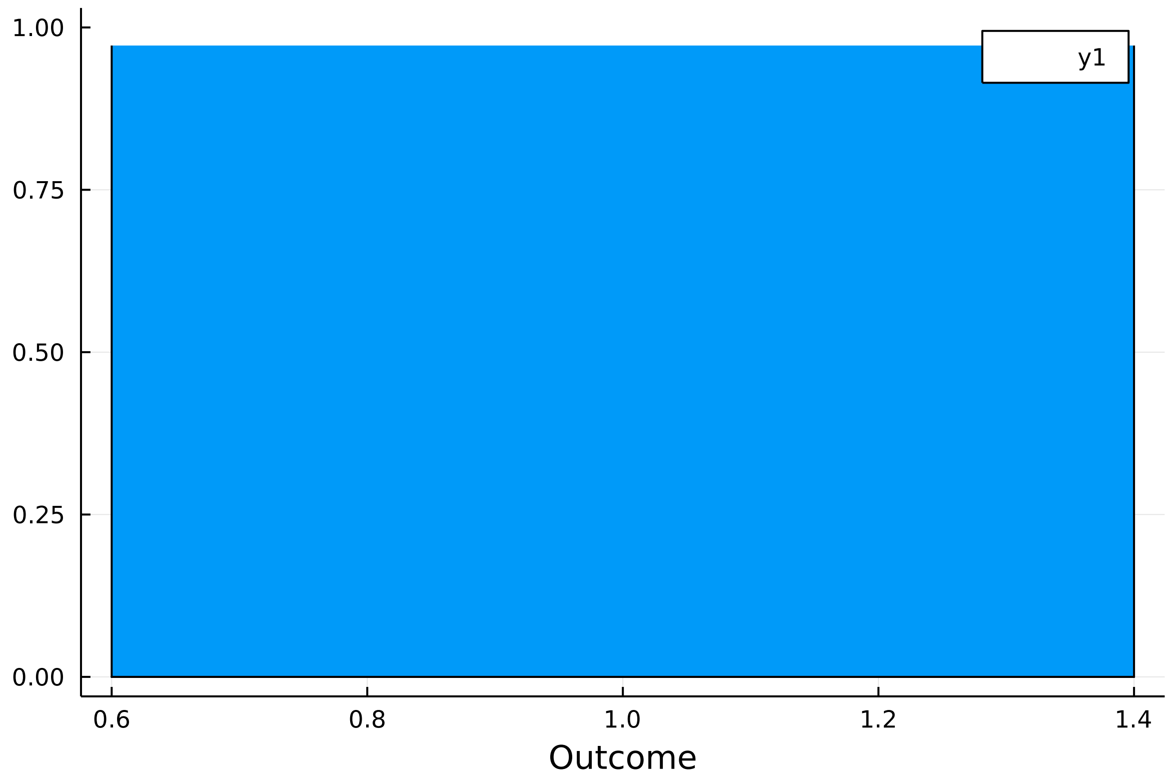
```
b8 = StatsPlots.boxplot(diabetes[!, :Age], xlabel = "Age")
```

Out[35]:



```
In [36]: b9 = StatsPlots.boxplot(diabetes[!, :Outcome], xlabel = "Outcome")
```

```
Out[36]:
```

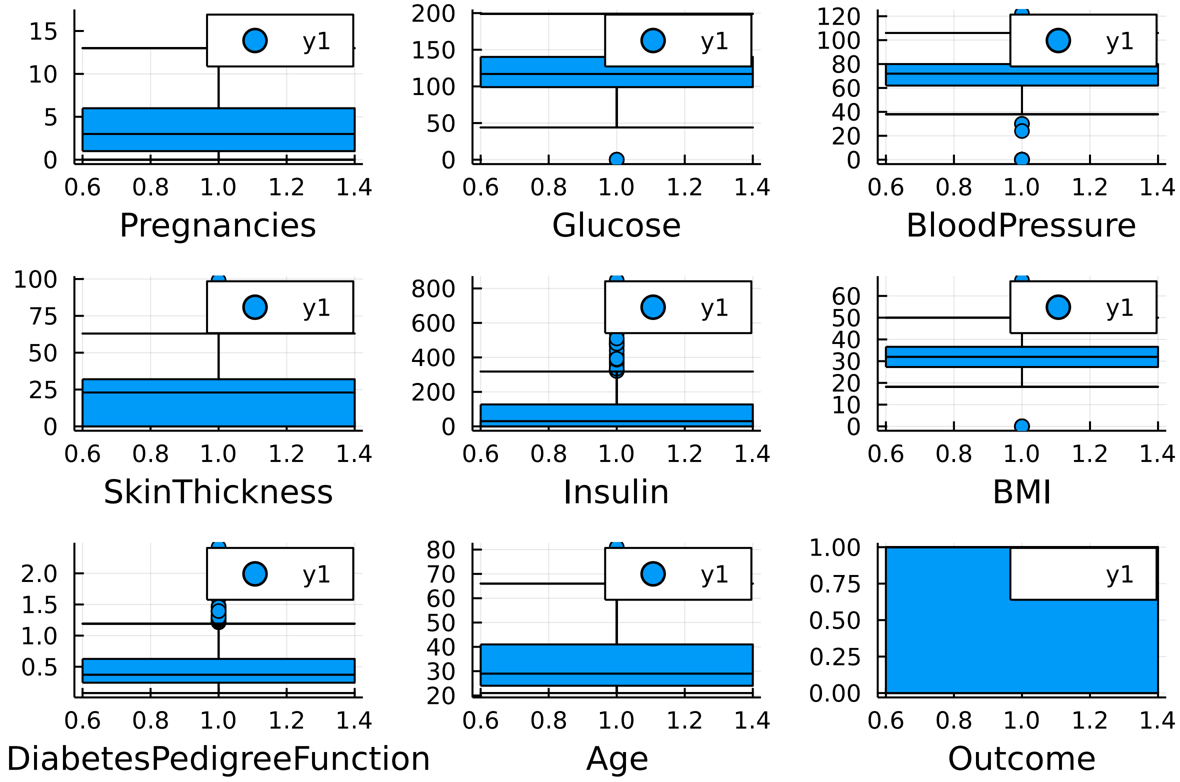


Stack Box Plot Together to Compare

In [37]:

```
plot!(b1, b2, b3, b4, b5, b6, b7, b8, b9)
```

Out[37]:

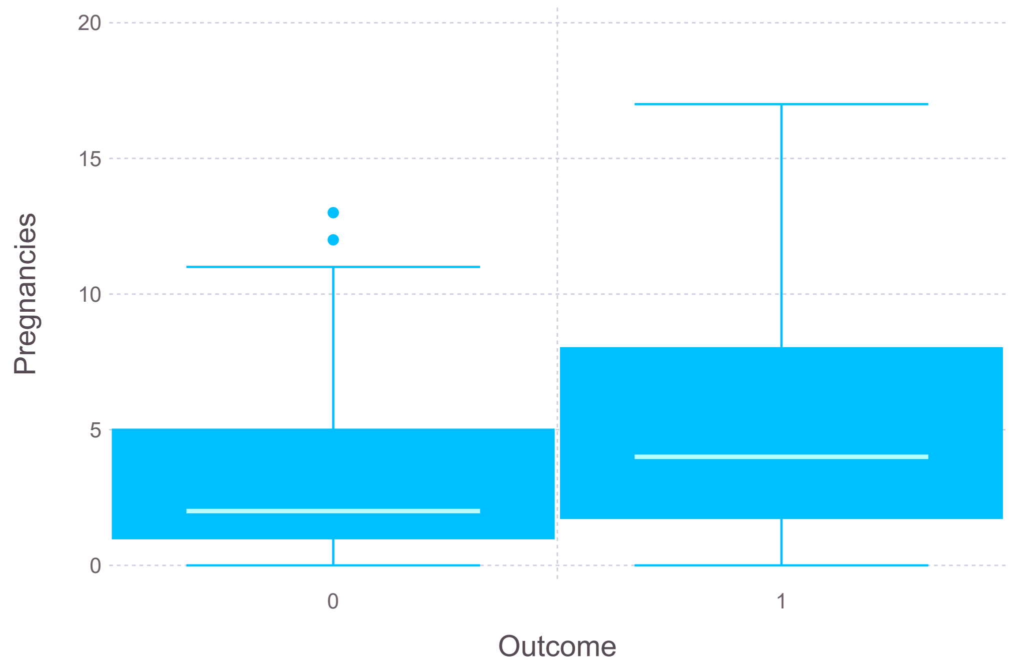


Side by Side Boxplots

- ### For **Pregnancies** variable there may have some outliers in the Category '0'

In [38]:
```
Gadfly.plot(diabetes, x=:Outcome, y=:Pregnancies, Geom.boxplot, Scale.x_discrete(levels=levels(diabetes.Outcome)))
```

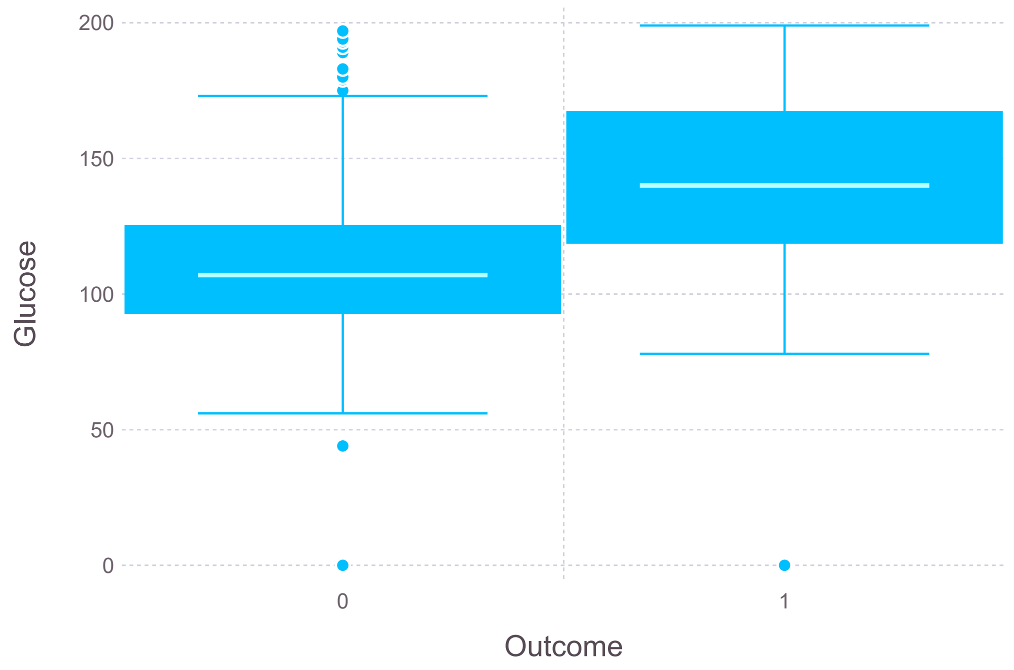
Out[38]:



- ### For *Glucose* variable there are numerous outliers in the Category '0', inspite of having smaller spread and little under Category '1'

In [39]:
```
Gadfly.plot(diabetes, x=:Outcome, y=:Glucose, Geom.boxplot, Scale.x_discrete(levels=levels(diabetes.Outcome)))
```

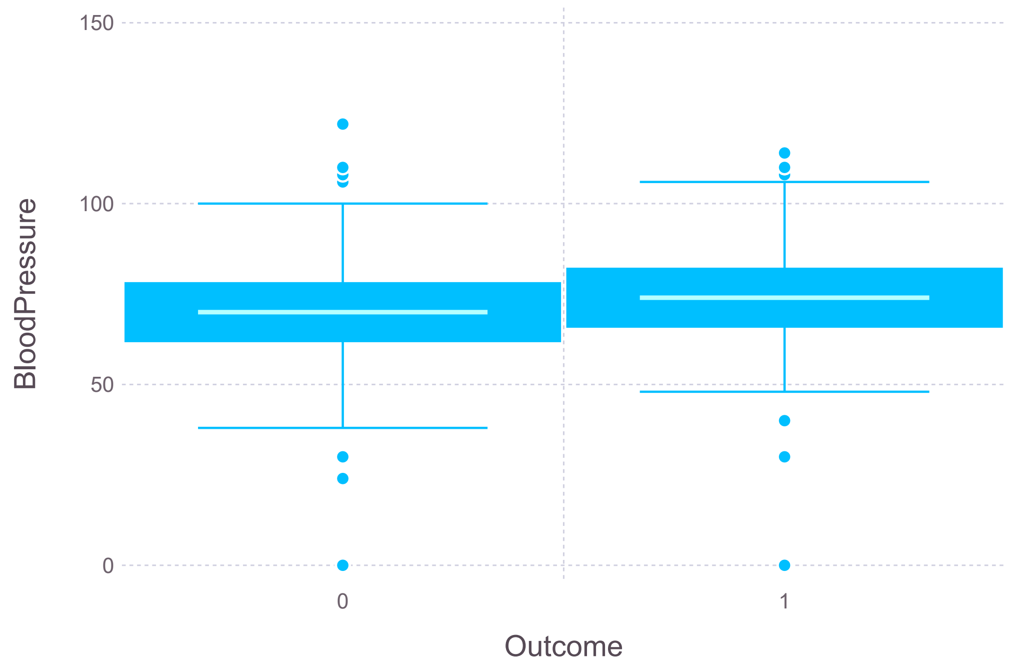
Out[39]:



- ### The Category '0' and Category '1' data for BloodPressure variable is almost similiar inspite of slight change.

In [40]:
```
Gadfly.plot(diabetes, x=:Outcome, y=:BloodPressure, Geom.boxplot, Scale.x_discrete(levels=levels(diabetes.Outcome)))
```

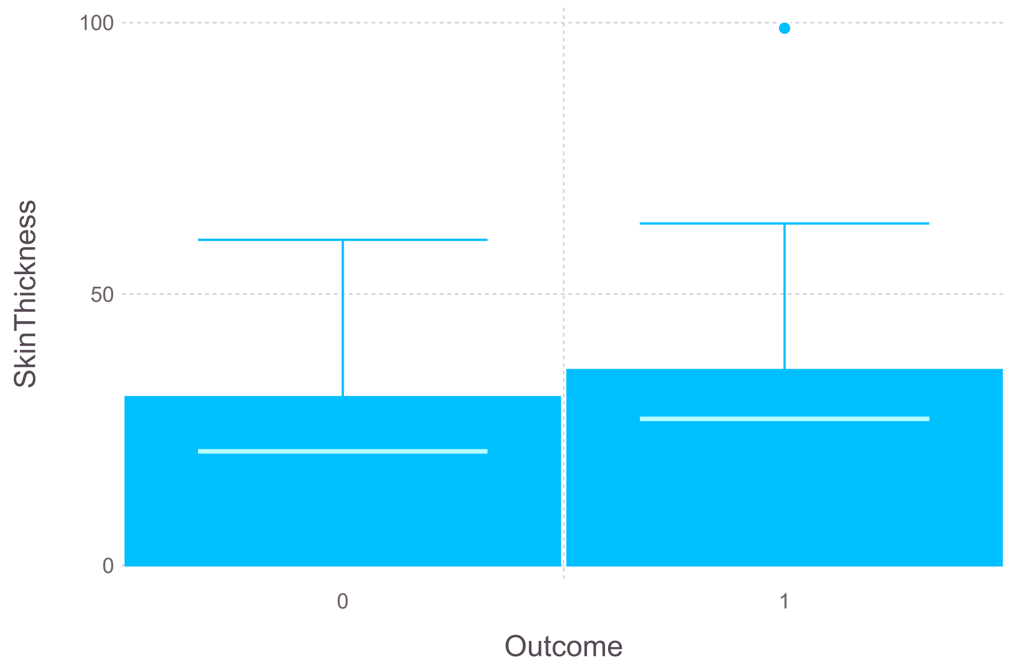
Out[40]:



- ### The Category '0' and Category '1' data for SkinThickness variable is almost similiar despite having little outliers

In [41]:
```
Gadfly.plot(diabetes, x=:Outcome, y=:SkinThickness, Geom.boxplot, Scale.x_discrete(levels=levels(diabetes.Outcome)))
```

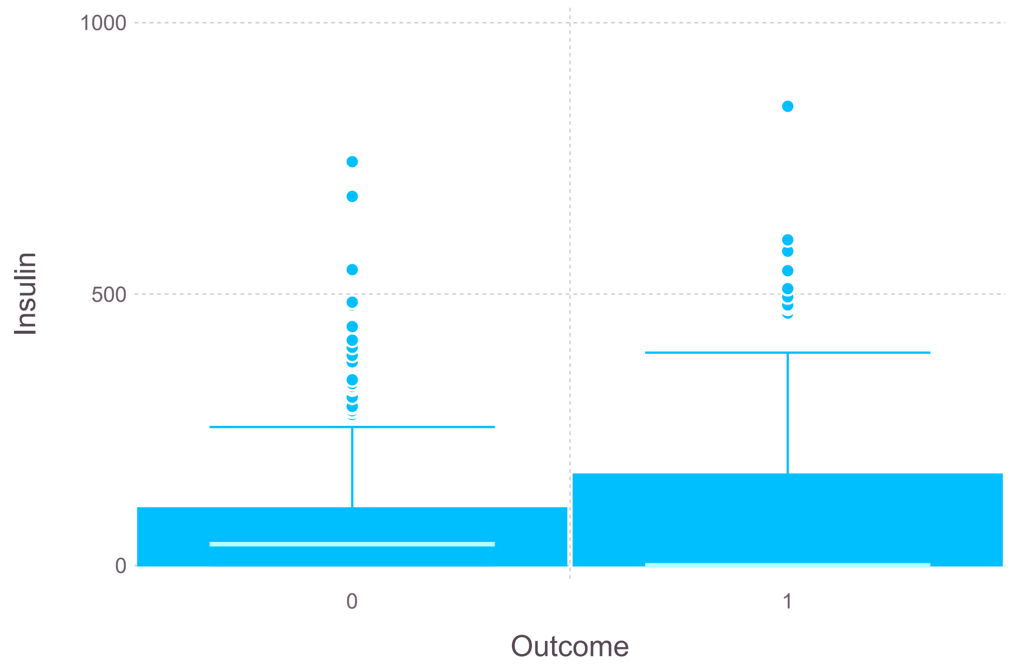
Out[41]:



- ### Both Category '0' and Category '1' there are numerous outliers for the variable Insulin. The spread is larger for the category '1'. The mean value is also deviate.

```
In [42]: Gadfly.plot(diabetes, x=:Outcome, y=:Insulin, Geom.boxplot, Scale.x_discrete(levels=levels(diabetes.Outcome)))
```

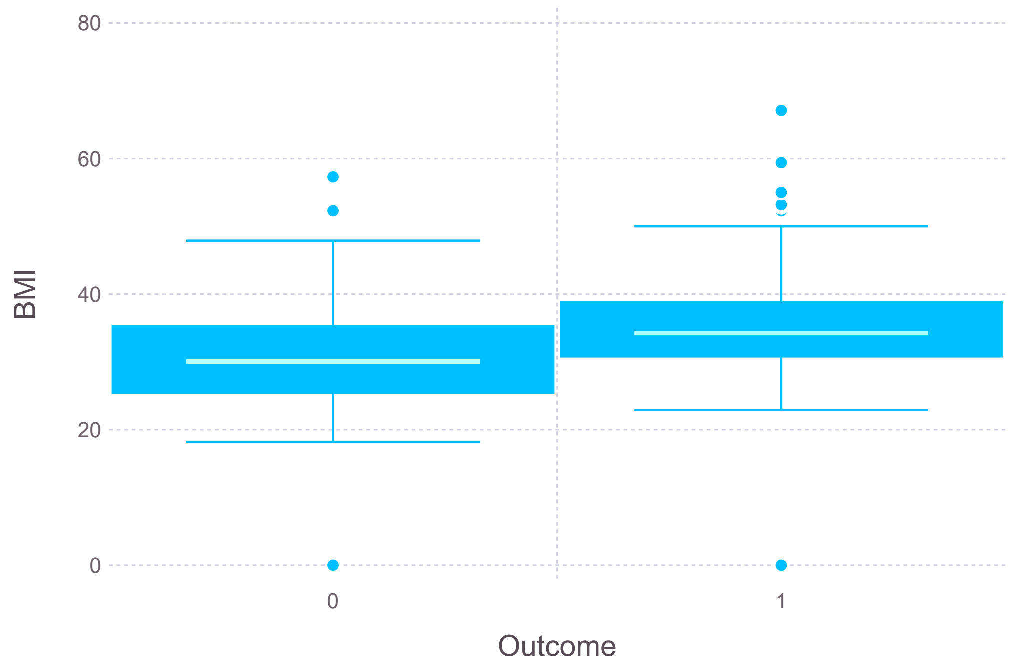
```
Out[42]:
```



- ### For the BMI, Category'1' have more outliers, although smaller spread and the center is also slightly deviate

In [43]:
```
Gadfly.plot(diabetes, x=:Outcome, y=:BMI, Geom.boxplot, Scale.x_discrete(levels=levels(diabetes.Outcome)))
```

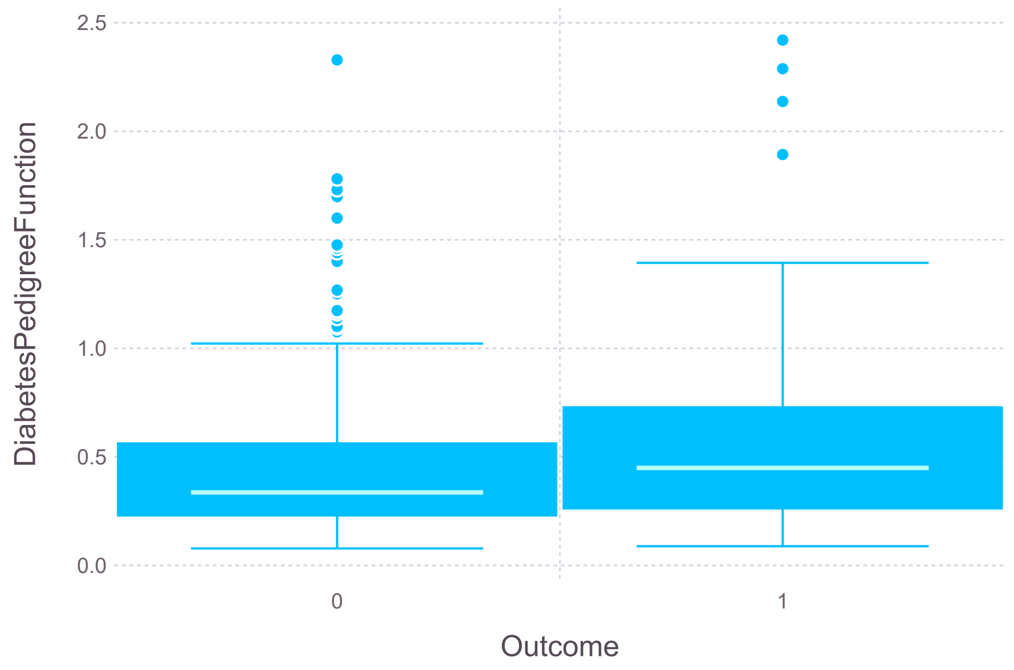
Out[43]:



- ### For DiabetesPedigreeFunction, the category '1' spread more than category'0', but having less outliers.

In [44]:
```
Gadfly.plot(diabetes, x=:Outcome, y=:DiabetesPedigreeFunction, Geom.boxplot, Scale.x_discrete(levels=levels(diabetes.Outc
```

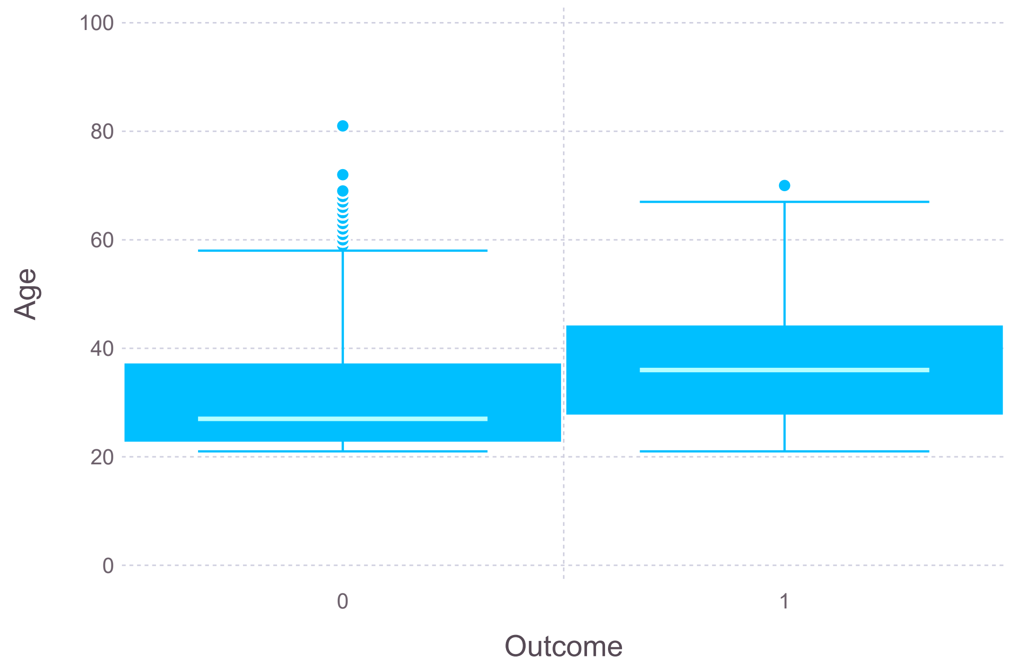
Out[44]:



- ### The Age values under the Category '0' have more outlier than other category

In [45]:
```
Gadfly.plot(diabetes, x=:Outcome, y=:Age, Geom.boxplot, Scale.x_discrete(levels=levels(diabetes.Outcome)))
```

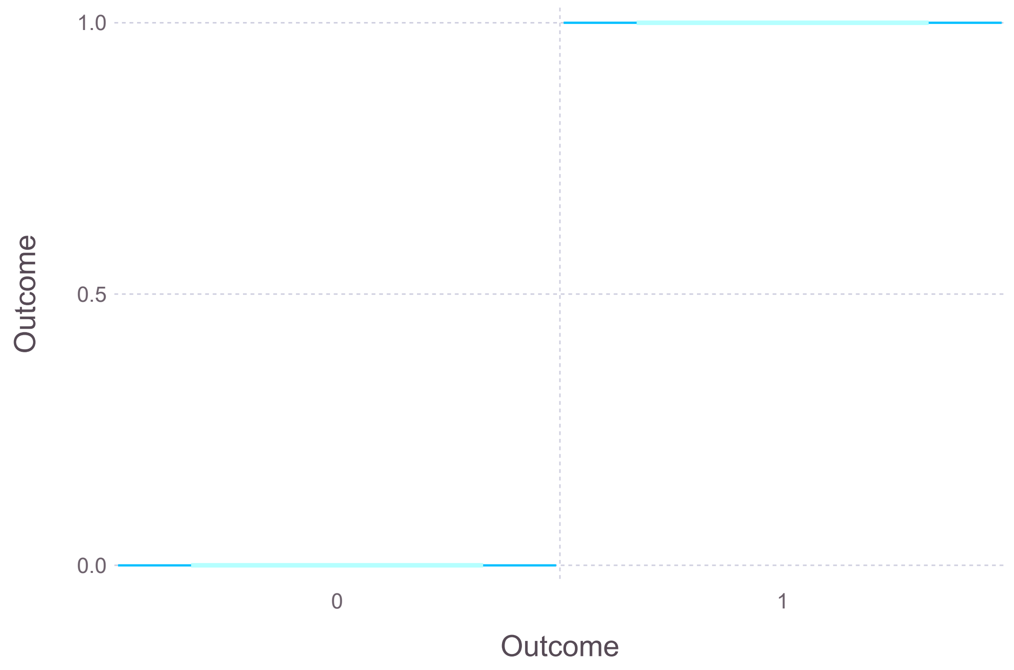
Out[45]:



- ### For the Categorical variable Outcome the distribution is uniform

In [46]:
```
Gadfly.plot(diabetes, x=:Outcome, y=:Outcome, Geom.boxplot, Scale.x_discrete(levels=levels(diabetes.Outcome)))
```

Out[46]:



```
In [ ]:
```