Визуалиция данных в Julia

Всем привет!
Этот тьюториал посвящен визуализации данных на языке программирования Julia (https://julialang.org/).

Глава 1: Gadfly

В данной части нашего тьюториала мы будем использовать пакет Gadfly (http://gadflyjl.org/dev/), который похож на ggplot в R.

Установка пакетов Gadfly, RDatasets, DataFrames

В Jupyter Notebook пакеты устанавливаются следующим образом:

In [ ]:

```
import Pkg
Pkg.add("Gadfly") #в кавычках - название необходимого пакета
```

```
   Updating registry at `C:\Users\arina\.julia\registries\General`
```

Установка занимает много времени, так что наберитесь терпения :)
Зато потом всё будет быстро!

Для проверки того, что пакет установился успешно, воспользуйтесь простым кодом:

```
using Gadfly
plot(y=[1,2,3])
```

В этом тьюториале мы будем использовать датасеты из пакета RDatasets (https://github.com/JuliaStats/RDatasets.jl) и функции библиотеки DataFrames (https://dataframes.juliadata.org/stable/), установите их тоже.

Начинаем с точечной диаграммы!

Во-первых, необходимо обратиться к установленным пакетам для дальнейшей работы с ними:

In [1]:

```
using Gadfly, RDatasets, DataFrames
```

Теперь сохраним в переменную `cars` датасет cars (https://rdrr.io/r/datasets/cars.html), в котором есть

данные о максимальной скорости (мили/час) и тормозного пути (футы) автомобилей в 1920-е.

In [2]:

```
cars = dataset("datasets", "cars")
```

Out[2]:

50 rows × 2 columns

	Speed	Dist
	Int64	Int64
1	4	2
2	4	10
3	7	4
4	7	22
5	8	16
6	9	10
7	10	18
8	10	26
9	10	34
10	11	17
11	11	28
12	12	14
13	12	20
14	12	24
15	12	28
16	13	26
17	13	34
18	13	34
19	13	46
20	14	26
21	14	36
22	14	60
23	14	80
24	15	20
25	15	26
26	15	54
27	16	32
28	16	40
29	17	32

	Speed	Dist
	Int64	Int64
30	17	40
⋮	⋮	⋮

В Gadfly есть основная функция `plot`, которая при работе с табличными данными (DataFrame (https://dataframes.juliadata.org/stable/)) имеет следующий синтаксис:

```
plot(data::AbstractDataFrame, elements::Element...; mapping...)
```

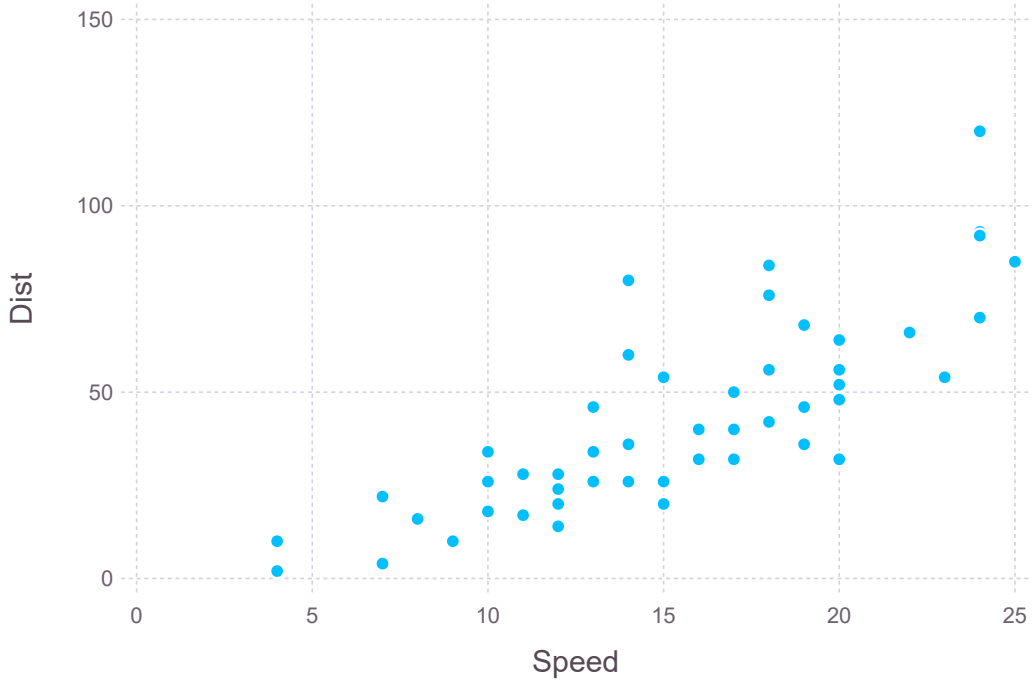
Первый аргумент - датасет, из которого берем данные для графика. Затем следуют элементы - столбцы из датасета для осей x, y и др. Последующие агрументы необходимы для обозначения вида графика (линейный, точечный, гистограмма, столбчатый) и его оформления.

Давайте сделаем **точечный график** взаимосвязи максимальной скорости и тормозного пути по датасету `cars`. По оси x будет столбец со скоростью `Speed`, y - тормозной путь `Dist`. `Geom.point` будет последним аргументом, чтобы график был точечным (если его не писать, то график все равно будет точечным).
Сохраним график в переменную `p_cars`:

In [3]:

```
p_cars = plot(cars, x=:Speed, y=:Dist, Geom.point)
```

Out[3]:

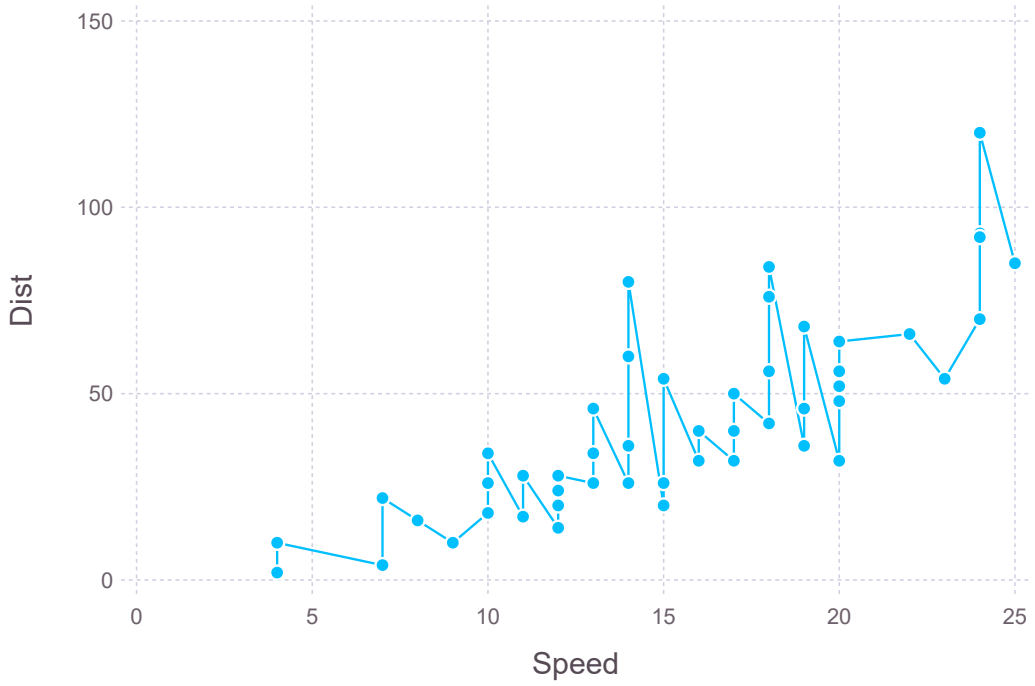


Также можно соединить точки на графике линиями, добавив `Geom.line`

In [4]:

```
plot(cars, x=:Speed, y=:Dist, Geom.point, Geom.line)
```

Out[4]:



Рассмотрим еще несколько возможностей на датасете [Motor Trend Car Road Tests (https://www.stat.auckland.ac.nz/~wild/data/Rdatasets/doc/datasets/mtcars.html)]

In [5]:

```
mtcars = dataset("datasets", "mtcars")
```

Out[5]:

32 rows × 12 columns (omitted printing of 4 columns)

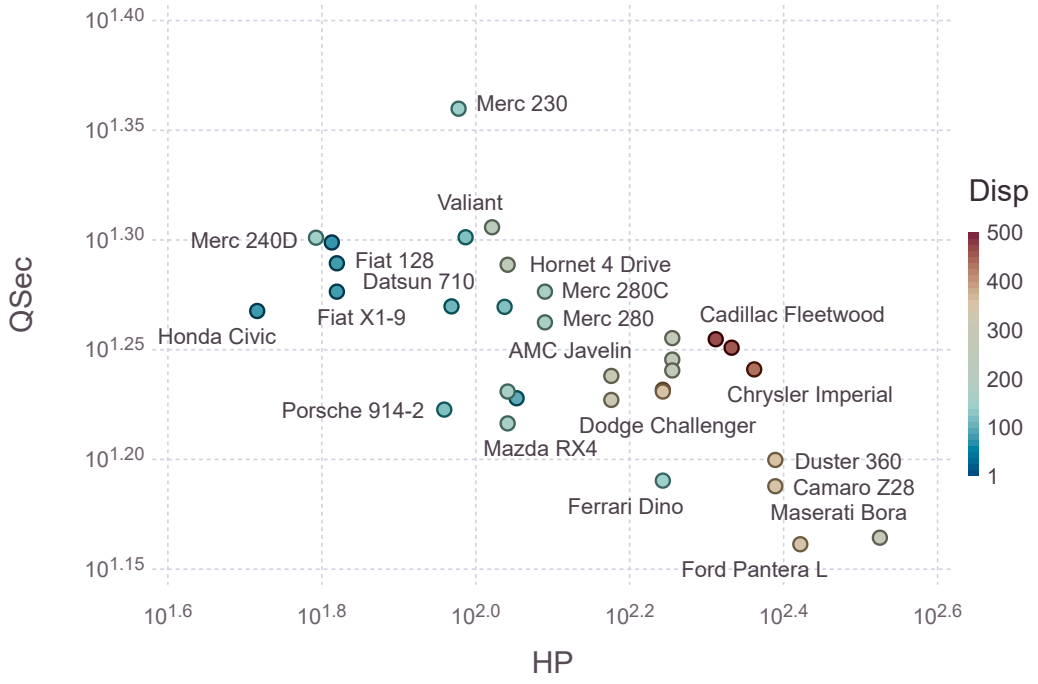
	Model	MPG	Cyl	Disp	HP	DRat	WT	QSec
	String	Float64	Int64	Float64	Int64	Float64	Float64	Float64
1	Mazda RX4	21.0	6	160.0	110	3.9	2.62	16.46
2	Mazda RX4 Wag	21.0	6	160.0	110	3.9	2.875	17.02
3	Datsun 710	22.8	4	108.0	93	3.85	2.32	18.61
4	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44
5	Hornet Sportabout	18.7	8	360.0	175	3.15	3.44	17.02
6	Valiant	18.1	6	225.0	105	2.76	3.46	20.22
7	Duster 360	14.3	8	360.0	245	3.21	3.57	15.84
8	Merc 240D	24.4	4	146.7	62	3.69	3.19	20.0

Добавим названией моделей на график, а также окрасим по значению Displacement. При помощи функции `Scale` можно менять масштаб осей для более удобного отображения данных

In [6]:

```
plot(mtcars, x=:HP, y=:QSec, label=:Model, color=:Disp,
        Geom.point, Geom.label, Scale.x_log10, Scale.y_log10)
```

Out[6]:



Линейный график

Выше мы рассмотрели, как создать **точечный график** с помощью аргумента `Geom.point` .
Для создания линейных, столбчатых графиков и гистограмм необходимо использовать другие аргументы.

Создадим **линейный график**.
Для этого необходимо использовать аргумент `Geom.line` .
Также для удобства возьмем новый датасет с экономическими показателями США (PCE - расходы на потребление в млрд долл., Pop - численность населения в тыс., PSavert - норма сбережения в %, UEmpMed - средняя продолжительность безработицы в неделях, Unemploy - численность безработных в тыс.):

In [7]:

```
data_economics = dataset("ggplot2", "economics")
```

Out[7]:

478 rows × 6 columns

	Date	PCE	Pop	PSavert	UEmpMed	Unemploy
	Date…	Float64	Int64	Float64	Float64	Int64
1	1967-06-30	507.8	198712	9.8	4.5	2944
2	1967-07-31	510.9	198911	9.8	4.7	2945
3	1967-08-31	516.7	199113	9.0	4.6	2958
4	1967-09-30	513.3	199311	9.8	4.9	3143
5	1967-10-31	518.5	199498	9.7	4.7	3066
6	1967-11-30	526.2	199657	9.4	4.8	3018
7	1967-12-31	532.0	199808	9.0	5.1	2878
8	1968-01-31	534.7	199920	9.5	4.5	3001
9	1968-02-29	545.4	200056	8.9	4.1	2877
10	1968-03-31	545.1	200208	9.6	4.6	2709
11	1968-04-30	550.9	200361	9.3	4.4	2740
12	1968-05-31	557.4	200536	8.9	4.4	2938
13	1968-06-30	564.4	200706	7.8	4.5	2883
14	1968-07-31	568.2	200898	7.6	4.2	2768
15	1968-08-31	569.5	201095	7.6	4.6	2686
16	1968-09-30	572.9	201290	7.8	4.8	2689
17	1968-10-31	578.0	201466	7.6	4.4	2715
18	1968-11-30	577.9	201621	8.1	4.4	2685
19	1968-12-31	584.9	201760	7.1	4.4	2718
20	1969-01-31	590.2	201881	6.5	4.9	2692
21	1969-02-28	590.4	202023	7.0	4.0	2712
22	1969-03-31	595.4	202161	6.6	4.0	2758
23	1969-04-30	601.8	202331	7.0	4.2	2713
24	1969-05-31	602.4	202507	7.9	4.4	2816
25	1969-06-30	604.3	202677	8.7	4.4	2868
26	1969-07-31	611.5	202877	8.5	4.4	2856
27	1969-08-31	614.9	203090	8.5	4.7	3040
28	1969-09-30	620.2	203302	8.3	4.5	3049
29	1969-10-31	622.1	203500	8.5	4.8	2856

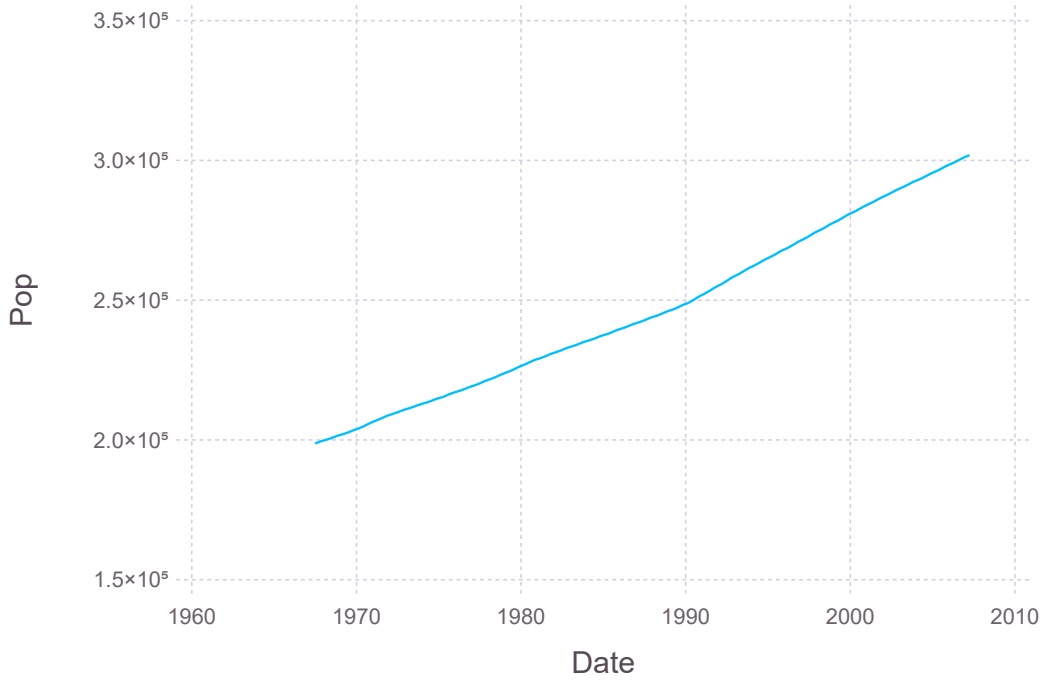
	Date	PCE	Pop	PSavert	UEmpMed	Unemploy
	Date…	Float64	Int64	Float64	Float64	Int64
30	1969-11-30	624.4	203675	8.6	4.6	2884
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Выведем график изменения численности населения во времени:

In [8]:

```
p_economics = plot(data_economics, x=:Date, y=:Pop, Geom.line)
```

Out[8]:



Теперь выведем на один график численность населения и количество безработных.
Для этого сохраним датасет только со столбцами `Date` , `Pop` , `Unemploy` в переменную `data_economics_` :

In [9]:

```
data_economics_ = dataset("ggplot2", "economics")[:,[:Date, :Pop, :Unemploy]]
```

Out[9]:

478 rows × 3 columns

	Date	Pop	Unemploy
	Date…	Int64	Int64
1	1967-06-30	198712	2944
2	1967-07-31	198911	2945
3	1967-08-31	199113	2958
4	1967-09-30	199311	3143
5	1967-10-31	199498	3066
6	1967-11-30	199657	3018
7	1967-12-31	199808	2878
8	1968-01-31	199920	3001

Кроме того, необходимо преобразовать нашу таблицу с помощью функции `stack` из `DataFrames`, которая создаст новые столбцы `variable` (со значениями Pop или Unemploy), `value` (численность населения или безработных):

In [10]:

```
println(stack(data_economics_,[:Pop,:Unemploy]))
```

```
956×3 DataFrame
│ Row │ Date       │ variable │ value  │
│     │ Dates.Date │ Cat…     │ Int64  │
├─────┼────────────┼──────────┼────────┤
│ 1   │ 1967-06-30 │ Pop      │ 198712 │
│ 2   │ 1967-07-31 │ Pop      │ 198911 │
│ 3   │ 1967-08-31 │ Pop      │ 199113 │
│ 4   │ 1967-09-30 │ Pop      │ 199311 │
│ 5   │ 1967-10-31 │ Pop      │ 199498 │
│ 6   │ 1967-11-30 │ Pop      │ 199657 │
│ 7   │ 1967-12-31 │ Pop      │ 199808 │
│ 8   │ 1968-01-31 │ Pop      │ 199920 │
│ 9   │ 1968-02-29 │ Pop      │ 200056 │
│ 10  │ 1968-03-31 │ Pop      │ 200208 │
│ 11  │ 1968-04-30 │ Pop      │ 200361 │
│ 12  │ 1968-05-31 │ Pop      │ 200536 │
│ 13  │ 1968-06-30 │ Pop      │ 200706 │
│ 14  │ 1968-07-31 │ Pop      │ 200898 │
│ 15  │ 1968-08-31 │ Pop      │ 201095 │
```

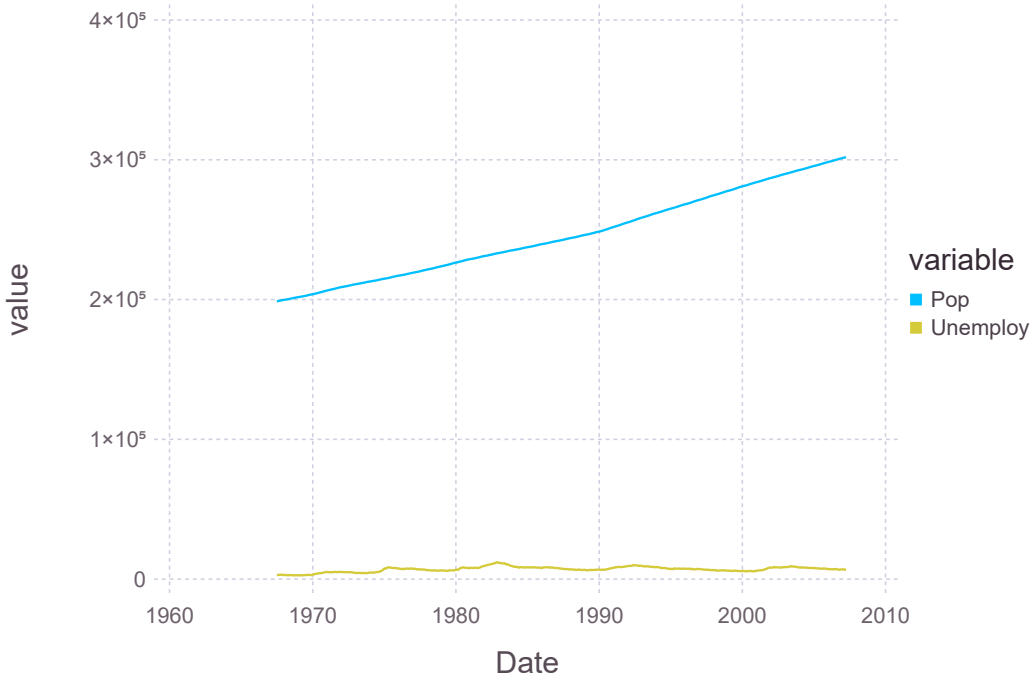
Подготовив нужным образом таблицу, теперь выведем численность населения и безработных, добавив

новый элемент `color` :

In [11]:

```
p_economics_ = plot(stack(data_economics_,[:Pop,:Unemploy]), x=:Date, y=:value, color=:vari
    Geom.line)
```

Out[11]:



Столбчатый график

Для создания **столбчатого** графика будем использовать `Geom.bar` .

Воспользуемся датасетом UCBAdmissions (https://rdrr.io/r/datasets/UCBAdmissions.html), в котором сохранены данные о поступающих в 6 основных департаментов Университет Беркли в 1973 году (Admit - статус "Поступил/Не поступил", Gender - пол, Dept - департамент, Freq - кол-во абитуриентов):

In [12]:

```
adm = dataset("datasets", "UCBAdmissions")
```

Out[12]:

24 rows × 4 columns

	Admit	Gender	Dept	Freq
	String	String	String	Int64
1	Admitted	Male	A	512
2	Rejected	Male	A	313
3	Admitted	Female	A	89
4	Rejected	Female	A	19
5	Admitted	Male	B	353
6	Rejected	Male	B	207
7	Admitted	Female	B	17
8	Rejected	Female	B	8

Сделаем новую таблицу, в которой не будет пола, следующим образом с помощью `combine` и `groupby` из `DataFrames` :

In [13]:

```
adm_n = combine(groupby(adm,[:Dept,:Admit]), :Freq=>sum=>:Frequency)
```

Out[13]:

12 rows × 3 columns

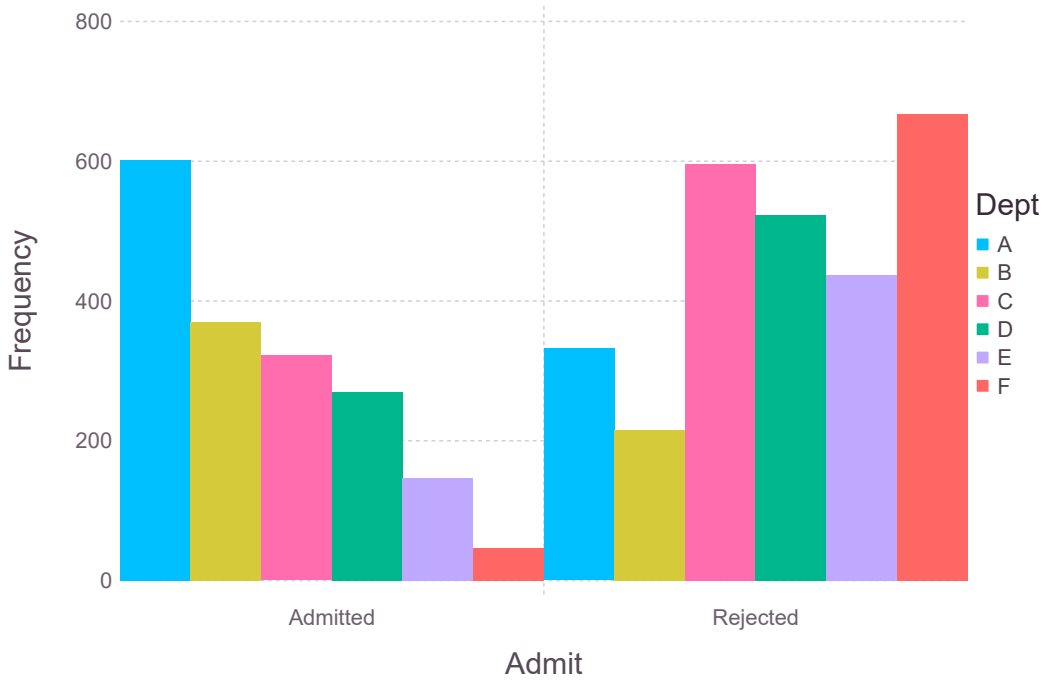
	Dept	Admit	Frequency
	String	String	Int64
1	A	Admitted	601
2	A	Rejected	332
3	B	Admitted	370
4	B	Rejected	215
5	C	Admitted	322
6	C	Rejected	596
7	D	Admitted	269
8	D	Rejected	523
9	E	Admitted	147
10	E	Rejected	437
11	F	Admitted	46
12	F	Rejected	668

Выведем **столбчатую диаграмму** с количеством поступивших и не поступивших по департаментам:

In [14]:

```
p_adm_1 = plot(adm_n, x=:Admit, y=:Frequency, color=:Dept, Geom.bar(position=:dodge))
```

Out[14]:



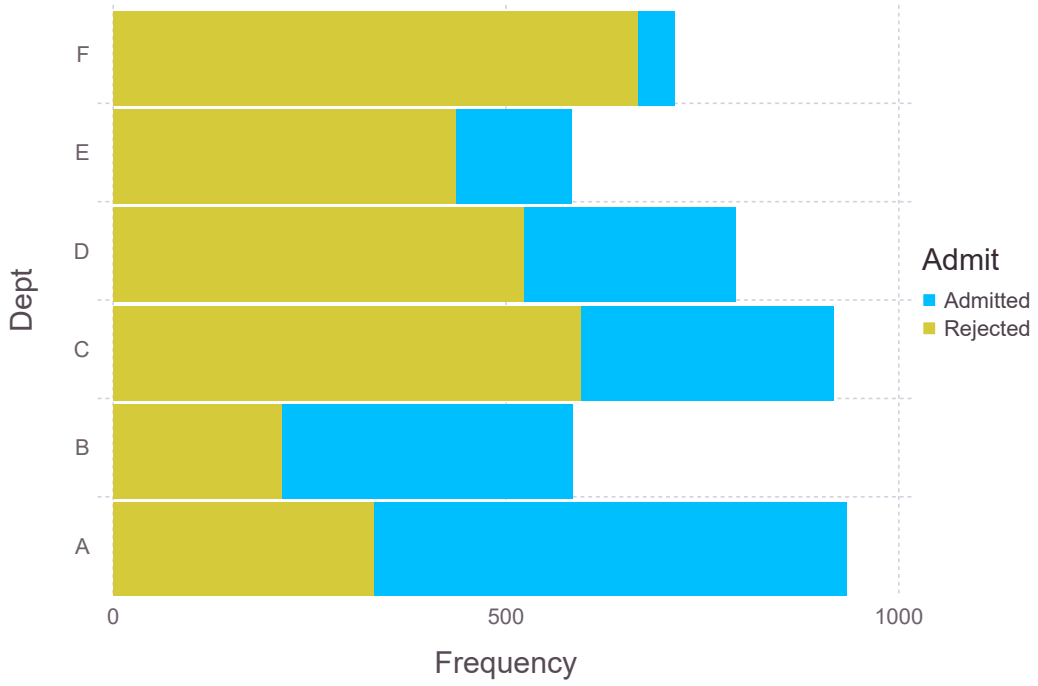
У `Geom.bar` могут быть два аргумента: `position` (*dodge* или *stack*), `orientation` (*vertical* или *horizontal*). У графика выше аргументы `Geom.bar(position=:dodge, orientation=:vertical)` .

Нарисуем график по департаментам с количеством поступивших и не поступивших с противоположными аргументами `Geom.bar(position=:stack, orientation=:horizontal)` . При смене ориентации графика необходимо данные для x и y поменять местами.

In [15]:

```
p_adm_2 = plot(adm_n, y=:Dept, x=:Frequency, color=:Admit,
    Geom.bar(position=:stack, orientation=:horizontal), Theme(bar_spacing= + 0.5mm))
```

Out[15]:



С помощью дополнительного аргумента `Theme(bar_spacing= + 0.5mm)` можно менять расстояние между столбцами в мм. При положительном числе столбцы будут находить на заданном расстоянии друг от друга, при отрицательном - пересекаться.

Гистограмма

Для создания **гистограммы**, которая отображает распределение случайной величины в выборке, будем использовать `Geom.histogram` .

Возмем большой датасет [diamonds (https://ggplot2.tidyverse.org/reference/diamonds.html)](https://ggplot2.tidyverse.org/reference/diamonds.html) с более 50 000 наблюдениями о стоимости, весе и других показателях бриллиантов:

In [16]:

```
diamonds = dataset("ggplot2", "diamonds")
```

Out[16]:

53,940 rows × 10 columns

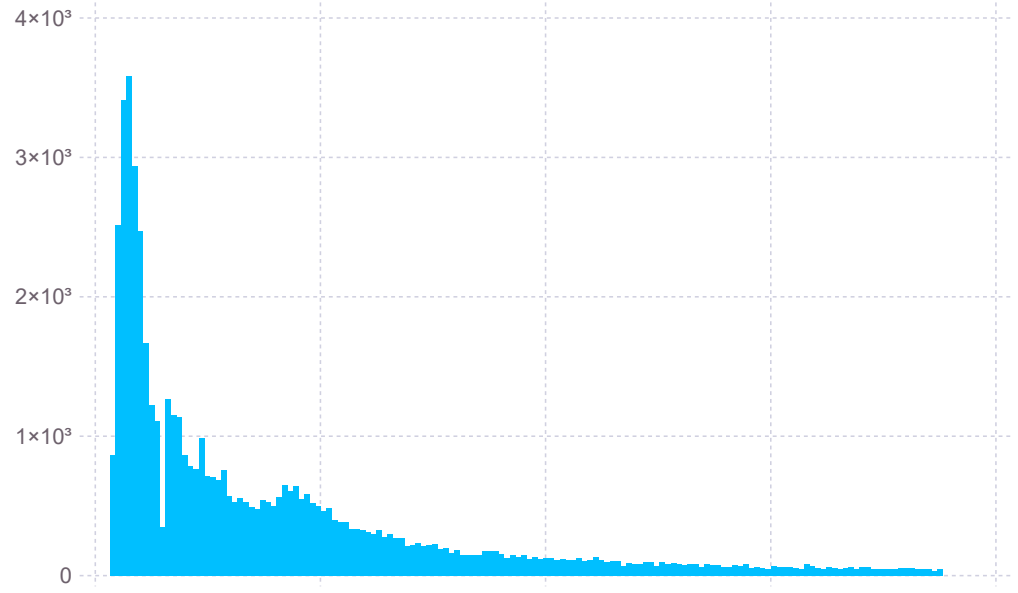
	Carat	Cut	Color	Clarity	Depth	Table	Price	X	Y	Z
	Float64	Cat…	Cat…	Cat…	Float64	Float64	Int32	Float64	Float64	Float64
1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
4	0.29	Premium	I	VS2	62.4	58.0	334	4.2	4.23	2.63
5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
6	0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48
7	0.24	Very Good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47
8	0.26	Very Good	H	SI1	61.9	55.0	337	4.07	4.11	2.53

Выведем распределение стоимости бриллиантов в датасете:

In [17]:

```
p_diamonds_1 = plot(diamonds, x=:Price, Geom.histogram)
```

Out[17]:



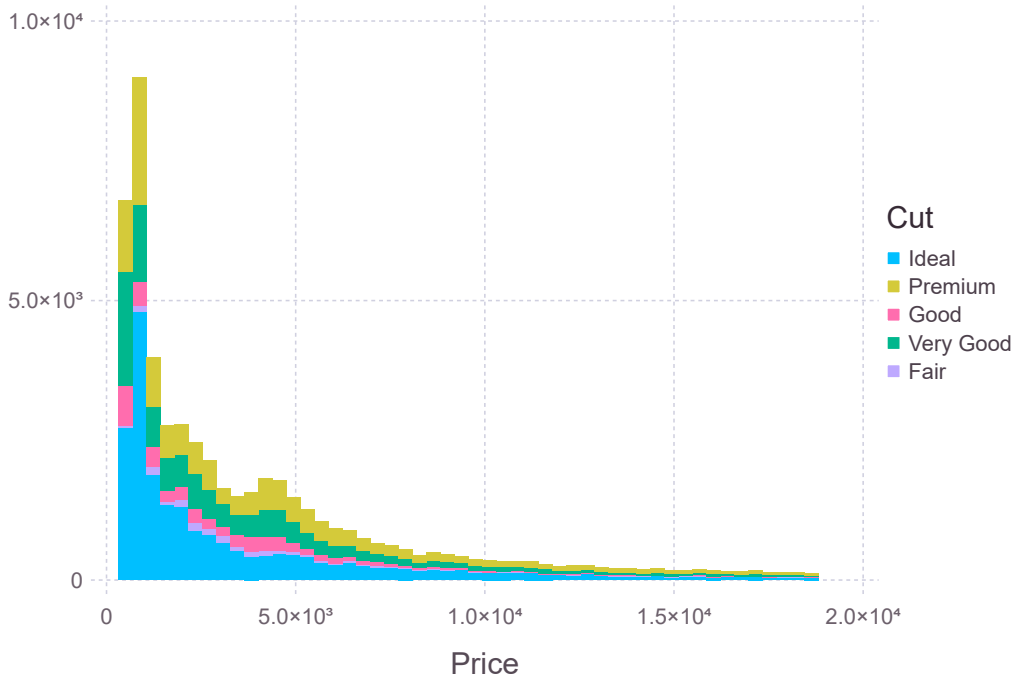
Также можно изменять толщину столбцов с помощью `Geom.histogram(bincount=50)` . Чем больше число, тем уже столбцы.

И добавим к предыдущему графику качество огранки.

In [18]:

```
p_diamonds_2 = plot(diamonds, x=:Price, color=:Cut, Geom.histogram(bincount=50))
```

Out[18]:



Совмещение графиков

Нарисуем два графика рядом

Для этого воспользуемся датасетом Smoking, Alcohol and (O)esophageal Cancer (https://www.stat.auckland.ac.nz/~wild/data/Rdatasets/doc/datasets/esoph.html)

In [19]:

```
data = dataset("datasets", "esoph")
```

Out[19]:

88 rows × 5 columns

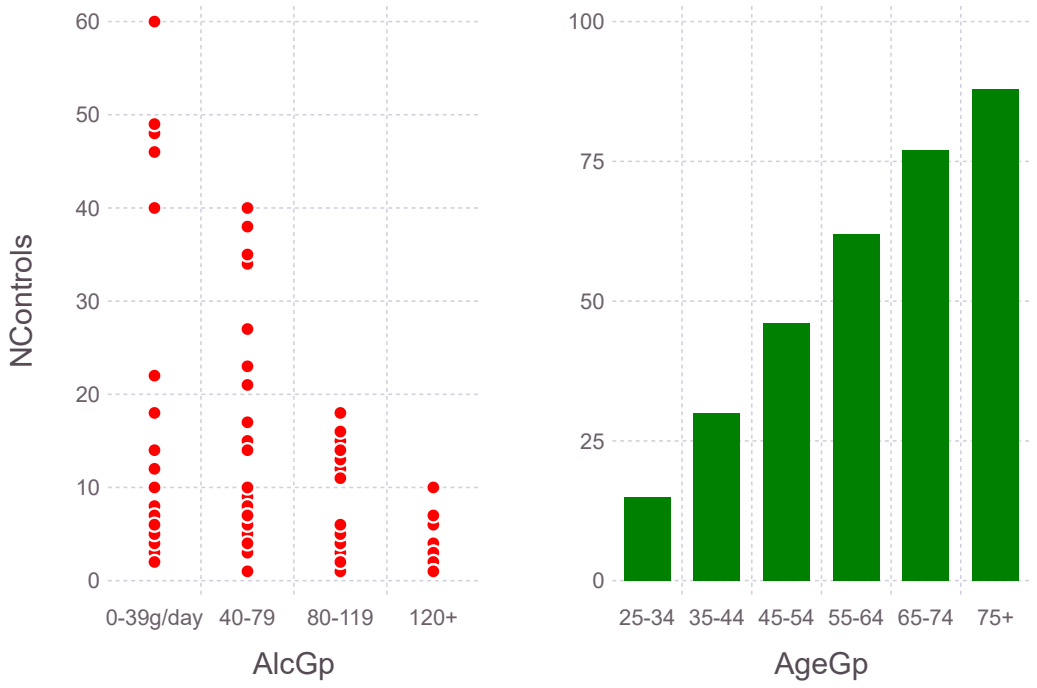
	AgeGp	AlcGp	TobGp	NCases	NControls
	Cat…	Cat…	Cat…	Int32	Int32
1	25-34	0-39g/day	0-9g/day	0	40
2	25-34	0-39g/day	10-19	0	10
3	25-34	0-39g/day	20-29	0	6
4	25-34	0-39g/day	30+	0	5
5	25-34	40-79	0-9g/day	0	27
6	25-34	40-79	10-19	0	7
7	25-34	40-79	20-29	0	4
8	25-34	40-79	30+	0	7

Графики можно расположить как горизонтально при помощи `hstack` ,

In [20]:

```
fig1a = plot(data, x="AlcGp", y="NControls", Geom.point, Theme(default_color=colorant"red")
fig1b = plot(data, x="AgeGp", Geom.bar, Theme(bar_spacing=0.3cm, default_color=colorant"gre
fig1 = hstack(fig1a, fig1b)
```

Out[20]:

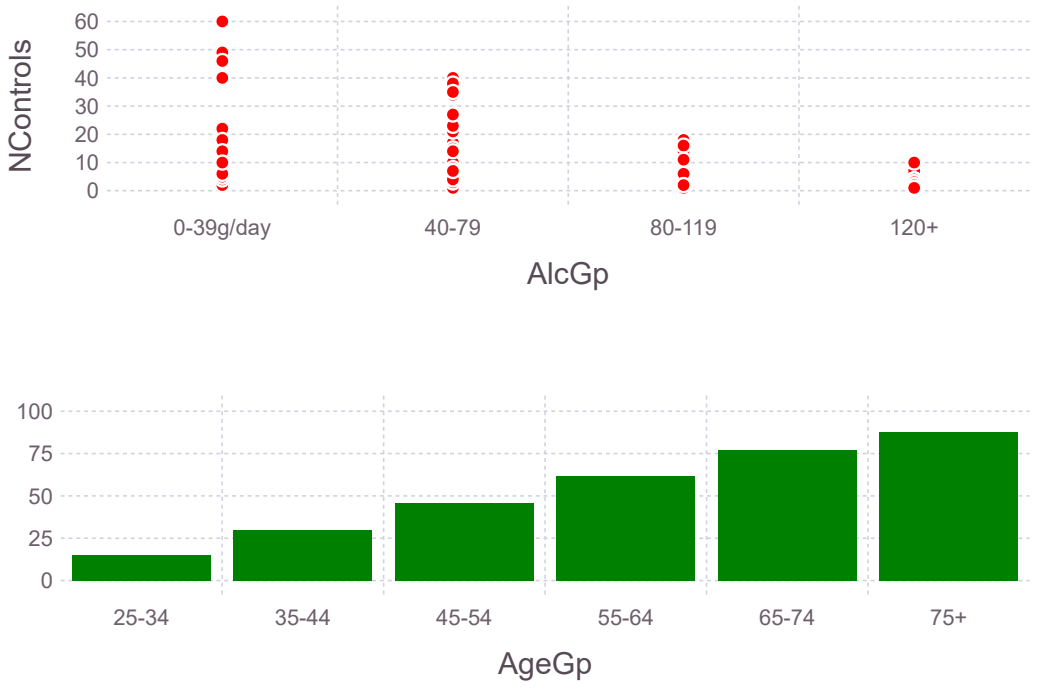


Так и вертикально при помощи функции `vstack`

In [21]:

```
fig1 = vstack(fig1a, fig1b)
```

Out[21]:



Построим два графика на одном при помощи функции `layers`

In [22]:

```
# Для начала создадим два датасета
points = DataFrame(index=rand(0:10,30), val=rand(1:10,30))
line = DataFrame(val=rand(1:10,11), index = collect(0:10))
```

Out[22]:

11 rows × 2 columns

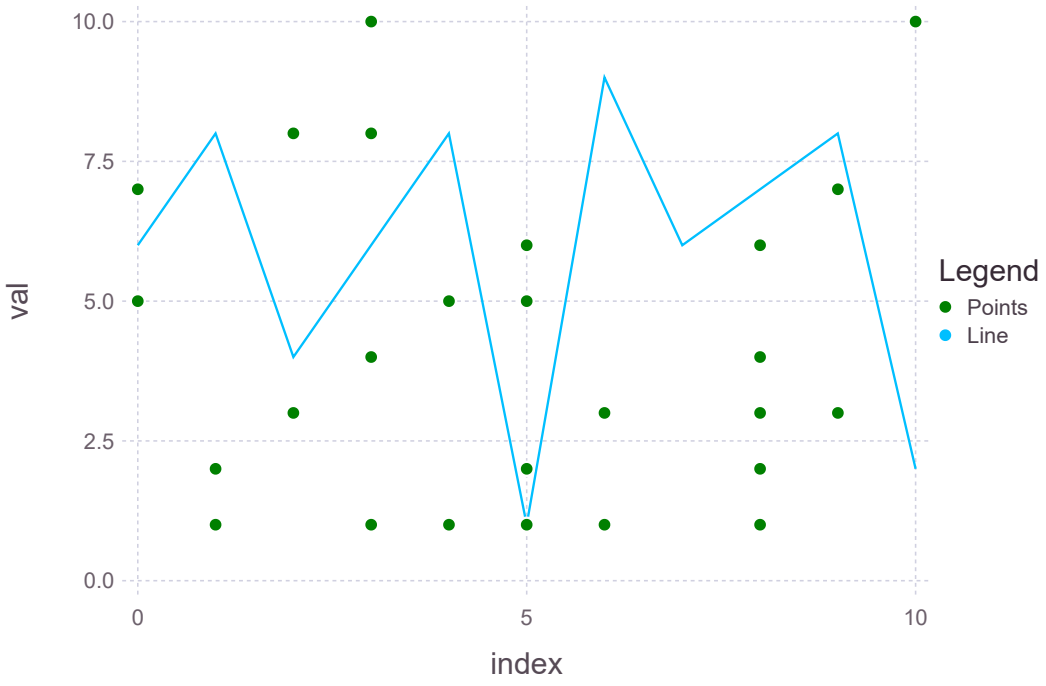
	val	index
	Int64	Int64
1	6	0
2	8	1
3	4	2
4	6	3
5	8	4
6	1	5
7	9	6
8	6	7

В каждом слое определяем, какой график хотим построить, и затем добавляем их в функцию `plot`

In [23]:

```
pointLayer = layer(points, x="index", y="val", Geom.point, Theme(default_color=colorant"gre
lineLayer = layer(line, x="index", y="val", Geom.line)
graph = plot(pointLayer, lineLayer, Guide.manual_color_key("Legend", ["Points", "Line"], ["
```

Out[23]:



Добавляем title и меняем название осей

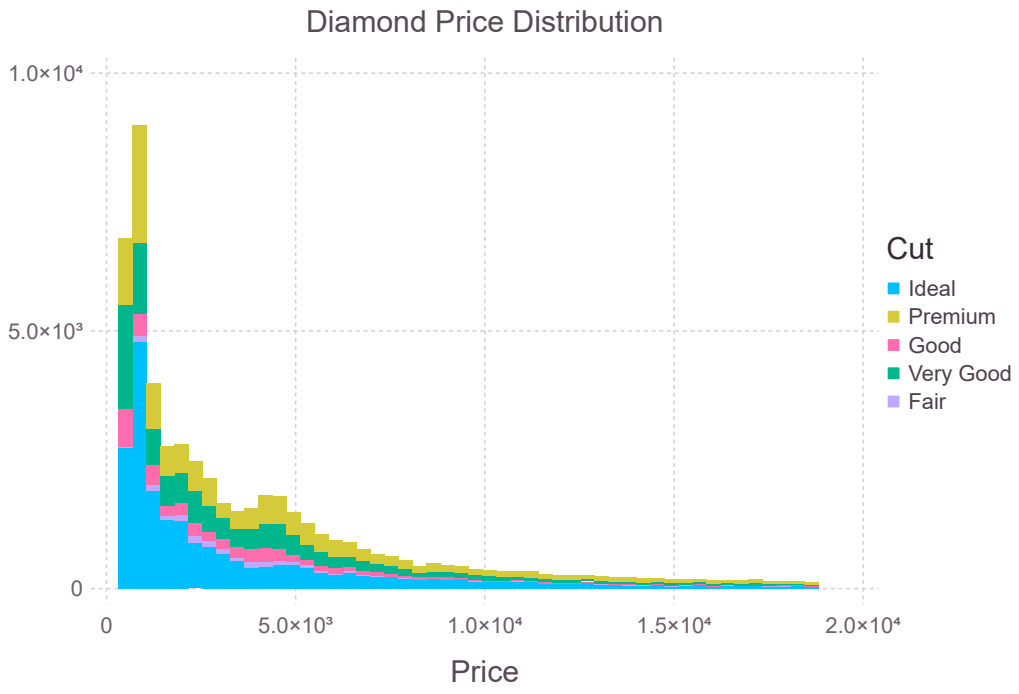
Для того, чтобы добавить название графика можно воспользоваться модулем `Guides` , очень схожий с модулем `Geometries` , который позволяет задавать тип графика

Возьмем один из ранее нарисованных графиков и добавим к нему `Guide.title`

In [24]:

```
p_diamonds_2 = plot(diamonds, x=:Price, color=:Cut, Geom.histogram(bincount=50), Guide.titl
```

Out[24]:

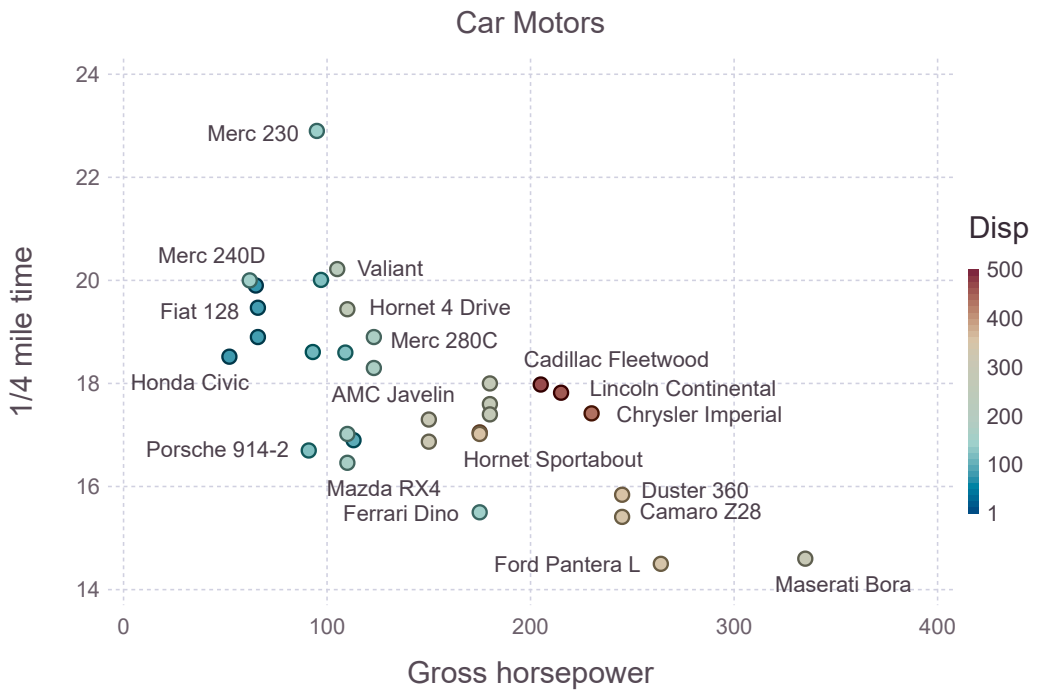


Модуль `Guides` также позволяет менять название осей.
Используем `Guide.xlabel` для оси Ox и `Guide.ylabel` для оси Oy

In [25]:

```
plot(mtcars, x=:HP, y=:QSec, label=:Model, color=:Disp,
        Geom.point, Geom.label, Guide.title("Car Motors"), Guide.ylabel("1/4 mile time"),
```

Out[25]:



В модуле `Guides` можно заменять метки на осях, цвета и еще несколько функций. Почитать об их применении более подробно можно здесь (http://gadflyjl.org/v0.6/lib/guides.html).

Бонус! С помощью Gadfly можно задавать тему

Наприемер, установим темную тему `Gadfly.push_theme(:dark)` . Вернуться к дефолтной тем можно при помощи `Gadfly.push_theme(:default)`

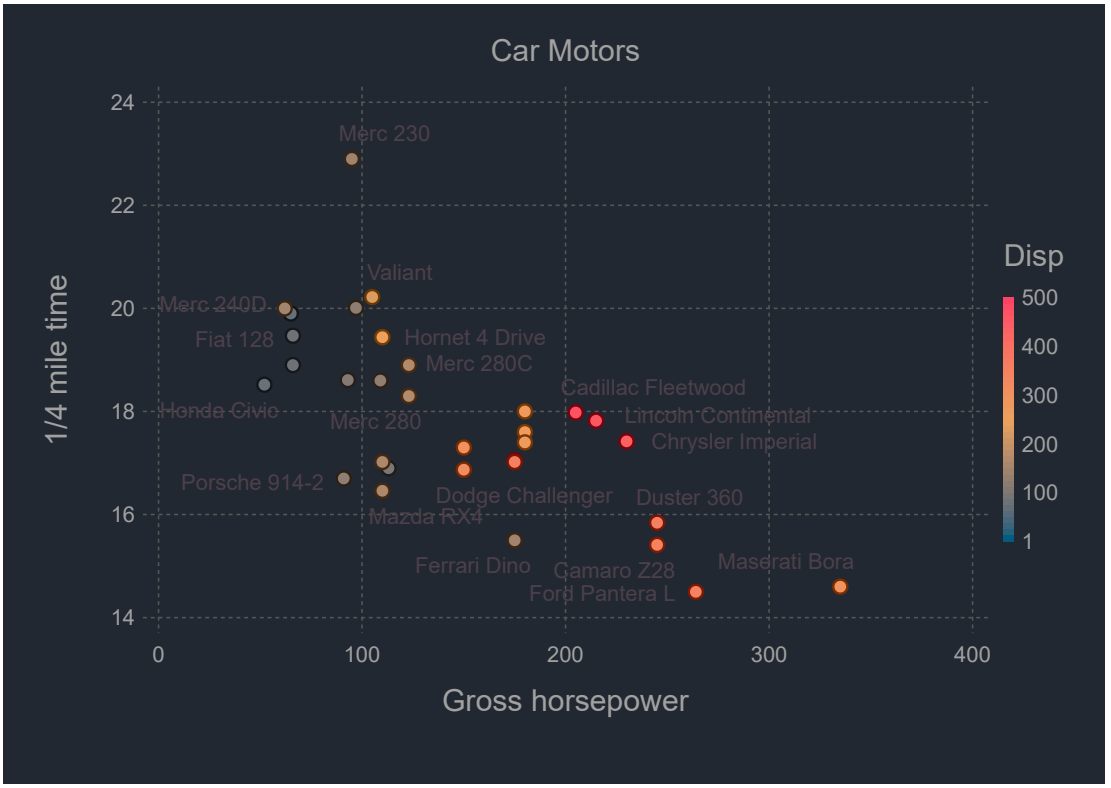
In [26]:

```
Gadfly.push_theme(:dark)
```

In [27]:

```
plot(mtcars, x=:HP, y=:QSec, label=:Model, color=:Disp,
        Geom.point, Geom.label, Guide.title("Car Motors"), Guide.ylabel("1/4 mile time"),
```

Out[27]:



Также можно задать тему самому!

In [28]:

```
black_panel = Theme(
    panel_fill="black",
    default_color="orange",
    point_size=1.5mm
)
```

Out[28]:

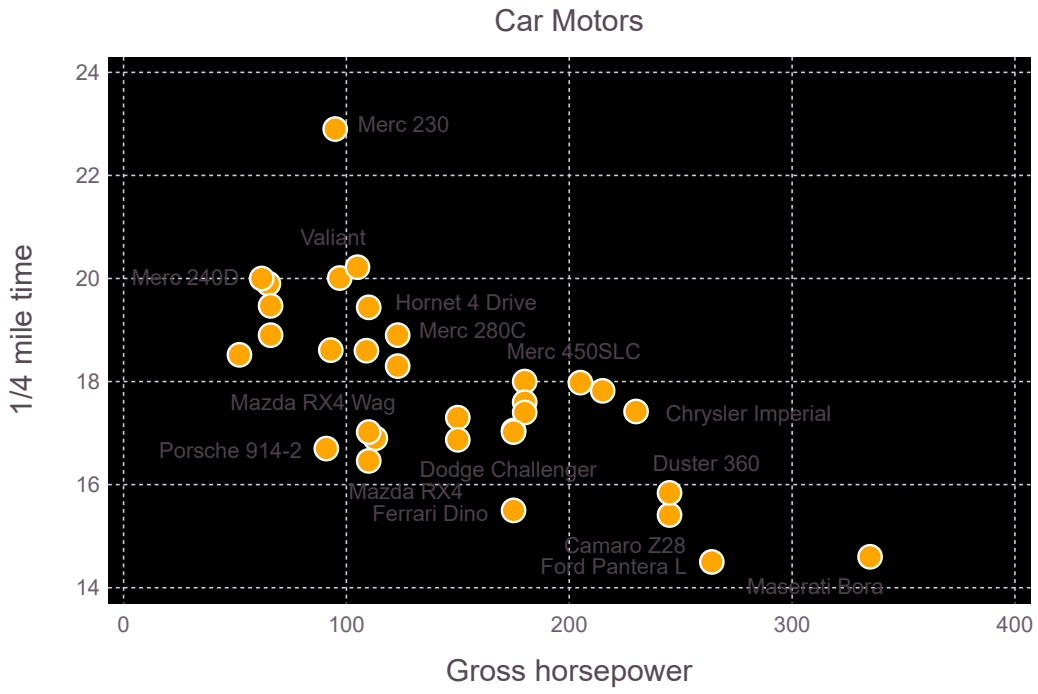
```
Theme(RGB{N0f8}(1.0,0.647,0.0), 1.5mm, 0.45mm, 1.8mm, Gadfly.Scale.default_di
screte_sizes, Gadfly.Scale.default_continuous_sizes, Gadfly.Scale.default_dis
crete_colors, Function[Compose.circle, Gadfly.Shape.square, Gadfly.Shape.diam
ond, Gadfly.Shape.cross, Gadfly.Shape.xcross, Gadfly.Shape.utriangle, Gadfly.
Shape.dtriangle, Gadfly.Shape.star1, Gadfly.Shape.star2, Gadfly.Shape.hexago
n, Gadfly.Shape.octagon, Gadfly.Shape.hline, Gadfly.Shape.vline, Gadfly.Shap
e.ltriangle, Gadfly.Shape.rtriangle], 0.3mm, [:solid, :dash, :dot, :dashdot,
:dashdotdot, :ldash, :ldashdash, :ldashdot, :ldashdashdot], [1.0, 0.9, 0.8,
0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.0], RGB{N0f8}(0.0,0.0,0.0), nothing, 0.3
mm, 0.0, nothing, Measures.Length{:mm,Float64}[5.0mm], RGB{N0f8}(0.816,0.816,
0.878), Measures.Length{:mm,Float64}[0.5mm, 0.5mm], RGB{N0f8}(0.627,0.627,0.6
27), 0.2mm, 0, "'PT Sans Caption','Helvetica Neue','Helvetica',sans-serif",
2.822222222222222mm, RGB{N0f8}(0.424,0.376,0.42), "'PT Sans','Helvetica Neu
e','Helvetica',sans-serif", 3.880555555555555mm, RGB{N0f8}(0.337,0.29,0.333),
"'PT Sans Caption','Helvetica Neue','Helvetica',sans-serif", 2.82222222222222
2mm, RGB{N0f8}(0.298,0.251,0.294), "'PT Sans','Helvetica Neue','Helvetica',sa
ns-serif", 3.880555555555555mm, RGB{N0f8}(0.212,0.165,0.208), "'PT Sans','Hel
vetica Neue','Helvetica',sans-serif", 2.822222222222222mm, RGB{N0f8}(0.298,0.
251,0.294), 40, -0.05mm, 1.0mm, 3.0mm, Gadfly.default_stroke_color, 0.3mm, Ga
dfly.default_discrete_highlight_color, Gadfly.default_continuous_highlight_co
lor, Gadfly.default_lowlight_color, Gadfly.default_middle_color, 0.6mm, :lef
t, :square, nothing, nothing, nothing, :right, nothing, 2.0mm, 1000, 10.0, 0.
5, 0.2, 1.0mm, 4, Gadfly.Scale.DiscreteColorScale(Gadfly.Scale.default_discre
te_colors, nothing, nothing, true), Gadfly.Scale.ContinuousColorScale(Gadfly.
Scale.var"#69#73"{Int64,Int64,Int64,Int64,Float64,Int64}(100, 40, 260, 10, 1.
5, 70), Gadfly.Scale.ContinuousScaleTransform(identity, identity, Gadfly.Scal
e.identity_formatter), nothing, nothing))
```

In [29]:

```
Gadfly.push_theme(black_panel)

plot(mtcars, x=:HP, y=:QSec, label=:Model,
         Geom.point, Geom.label, Guide.title("Car Motors"), Guide.ylabel("1/4 mile time"),
```

Out[29]:



Подробнее о стилизации графиков можно прочитать [здесь](http://gadflyjl.org/v0.6.3/man/themes.html#The-Dark-theme-1)

Сохранение графиков

Если необходимо сохранить график в виде картинки, то воспользуйтесь следующим кодом:

```
img = SVG("cars_plot.svg", 20cm, 10cm) #аргументы - название, ширина, длина картин
ки
draw(img, p_cars) # p_cars - график, созданный раннее
```

Так картинка будет в формате .svg, расположенная в папке с Вашим кодом.

In [30]:

```
println(RDatasets.datasets("ggplot2")) #часть доступных датасетов
```

```
8×5 DataFrame
| Row | Package | Dataset      | Title
| Rows  | Columns |
|     | String  | String       | String
| Int64 | Int64   |
|-----|---------|--------------|-----------------------------------------------
-----------------------|---------|---------|
| 1   | ggplot2 | diamonds     | Prices of 50,000 round cut diamonds
| 53940 | 10      |
| 2   | ggplot2 | economics    | US economic time series.
| 478   | 6       |
| 3   | ggplot2 | midwest      | Midwest demographics.
| 437   | 28      |
| 4   | ggplot2 | movies       | Movie information and user ratings from IMD
B.com.                  | 58788 | 24      |
| 5   | ggplot2 | mpg          | Fuel economy data from 1999 and 2008 for 38
 popular models of car | 234   | 11      |
| 6   | ggplot2 | msleep       | An updated and expanded version of the mamma
ls sleep dataset.     | 83    | 11      |
| 7   | ggplot2 | presidential | Terms of 10 presidents from Eisenhower to Bu
sh W.                 | 10    | 4       |
| 8   | ggplot2 | seals        | Vector field of seal movements.
| 1155  | 4       |
```