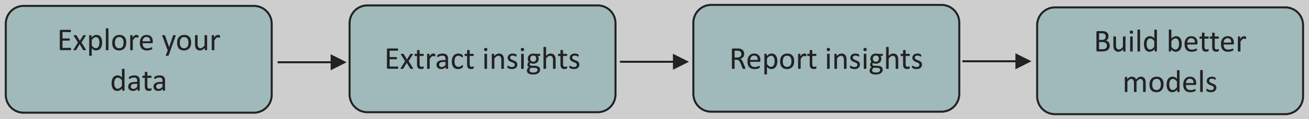
Creating advanced plots with `ggplot2`

Data visualization in R

- Data visualization is an essential part in the data analysis process:



- R's plotting packages enable customized graphs and more:

same as last unit

Packages	Description
Base R Graphics/grDevices	Built-in plotting functionalities in R base
Ggplot2	"Grammar of Graphics": build your plot from various layers
Lattice	Provides functionalities for producing Trellis graphics
Plotly	Create Interactive Web Graphics via "plotly.js"

https://wiki.python.org/moin/NumericAndScientific/Plotting

Why use `ggplot2` instead of Base R

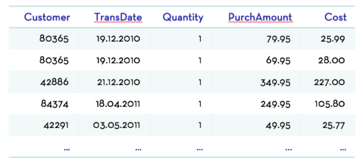
- `ggplot2` is based on the "Grammar of Graphics":
  - Provides a schema for data visualization by breaking up graphs into semantic components such as scales and layers.

```
myPlot <- ggplot()+ geom_point()+ ...
```

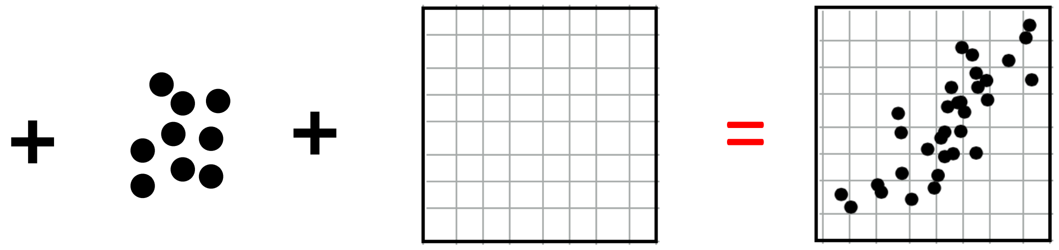
Alternatively, add the layers step by step to a variable: 1

```
myPlot <- ggplot ()
myPlot <- myPlot + geom_point()
```

  - Used to create more flexible plots than in Base R.



Customer	TransDate	Quantity	PurchAmount	Cost
80365	19.12.2010	1	79.95	25.99
80365	19.12.2010	1	69.95	28.00
42886	21.12.2010	1	349.95	227.00
84374	18.04.2011	1	249.95	105.80
42291	03.05.2011	1	49.95	25.77
...	...	...	...	...



data

geoms

coordinate system

plot

Data: data set and variables 2

Geoms: Visual marks to represent the data points 3

Coordinate system: location to place the points 4

Why use `ggplot2` instead of Base R

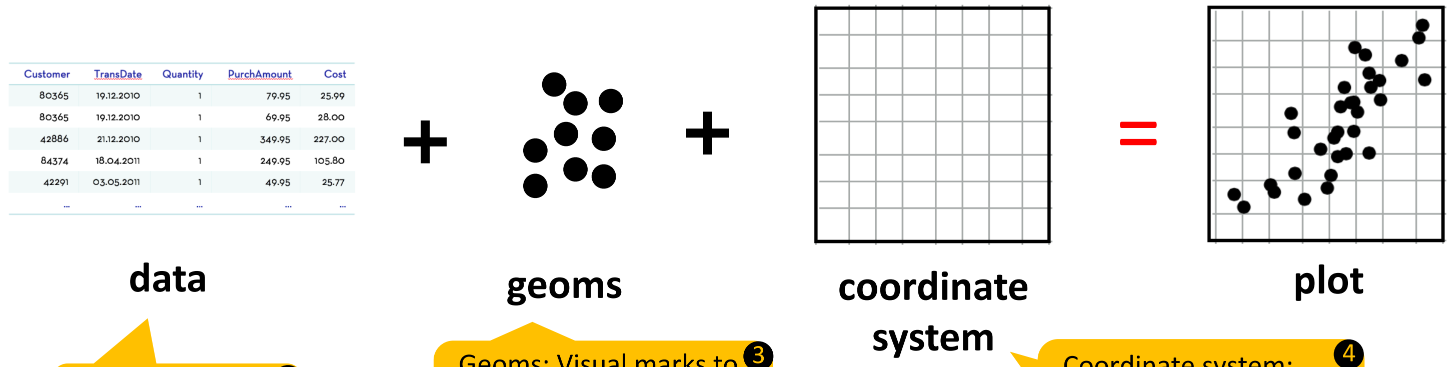
- `ggplot2` is based on the "Grammar of Graphics":
  - Provides a schema for data visualization by breaking up graphs into semantic components such as scales and layers.

```
myPlot <- ggplot()+ geom_point()+ ...
```

1 Alternatively, add the layers step by step to a variable:

```
myPlot <- ggplot ()
myPlot <- myPlot + geom_point()
```

  - Used to create more flexible plots than in Base R.



Customer	TransDate	Quantity	PurchAmount	Cost
80365	19.12.2010	1	79.95	25.99
80365	19.12.2010	1	69.95	28.00
42886	21.12.2010	1	349.95	227.00
84374	18.04.2011	1	249.95	105.80
42291	03.05.2011	1	49.95	25.77
...	...	...	...	...

2 Data: data set and variables

3 Geoms: Visual marks to represent the data points

4 Coordinate system: location to place the points

Why use `ggplot2` instead of Base R

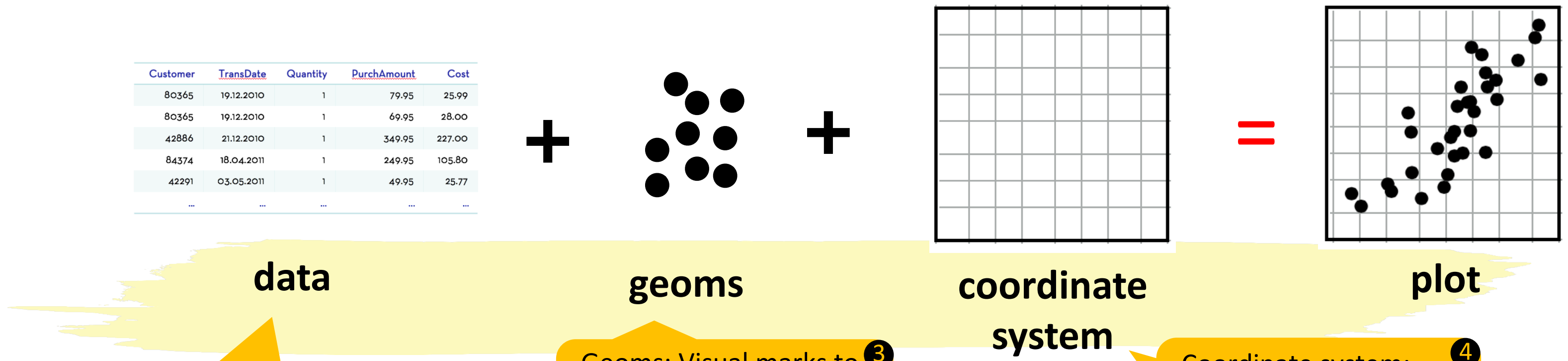
- `ggplot2` is based on the "Grammar of Graphics":
    - Provides a schema for data visualization by breaking up graphs into semantic components such as scales and layers.

```
myPlot <- ggplot()+ geom_point()+ ...
```

1 Alternatively, add the layers step by step to a variable:

```
myPlot <- ggplot ()
myPlot <- myPlot + geom_point()
```

- Used to create more flexible plots than in Base R.



Customer	TransDate	Quantity	PurchAmount	Cost
80365	19.12.2010	1	79.95	25.99
80365	19.12.2010	1	69.95	28.00
42886	21.12.2010	1	349.95	227.00
84374	18.04.2011	1	249.95	105.80
42291	03.05.2011	1	49.95	25.77
...	...	...	...	...

2 Data: data set and variables

3 Geoms: Visual marks to represent the data points

4 Coordinate system: location to place the points

How to plot
Steps

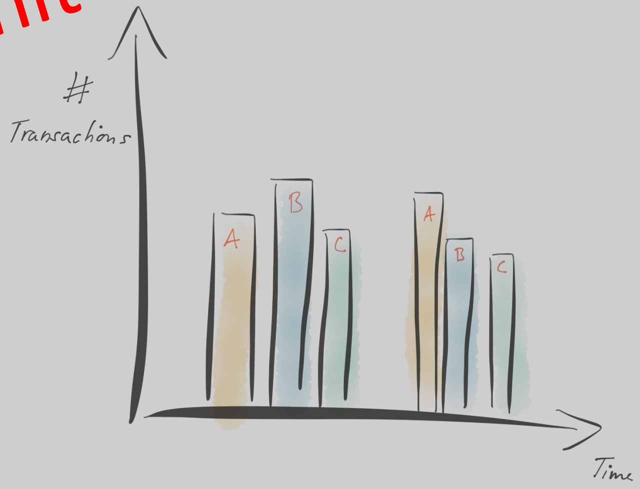
1. Choose the plot type
2. Find the appropriate R function
3. Transform data
4. Create the plot
5. Improve aesthetic features of the plot
6. Save plot

same as last unit

Step 1: Choose the plot type
Decide the best way to convey the information

- What do you want to show?
  - A single variable?
  - The relationship between multiple variables?
- Is your data continuous or discrete?

same as last unit

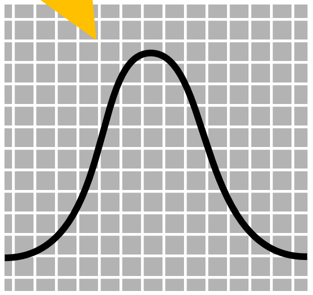


Step 2: Find the function
Plotting a single variable

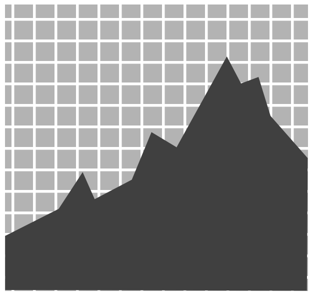
Kernel density estimator and histogram

Continuous

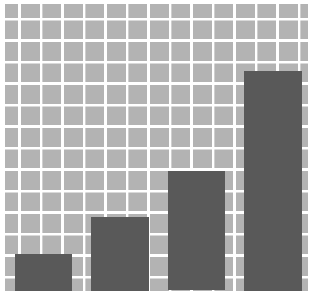
Discrete



`geom_density()`



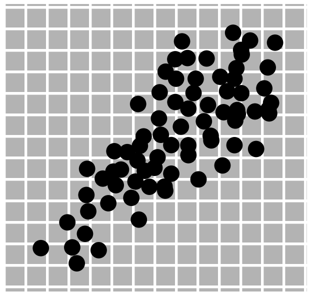
`geom_area()`

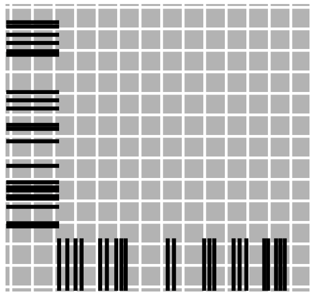


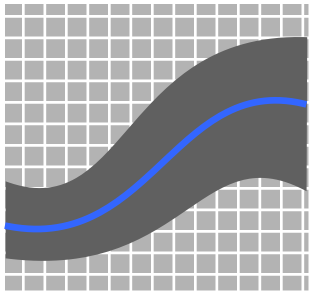
`geom_bar()`

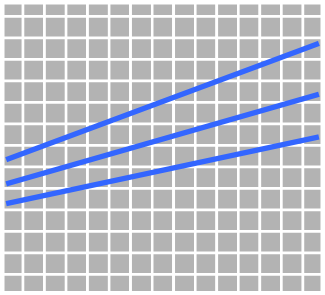
Step 2: Find the function
Plotting two variables

Continuous Continuous

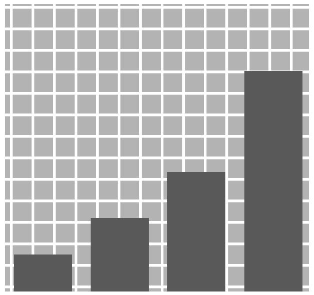

`geom_point()`

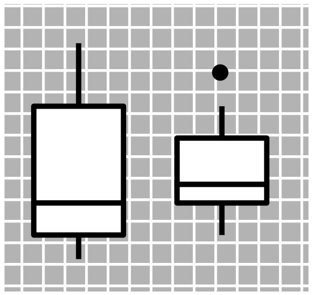

`geom_rug()`

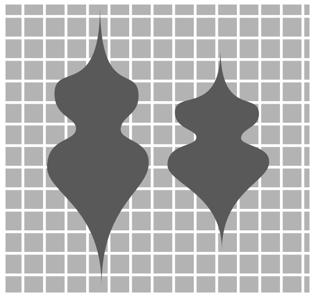

`geom_smooth()`

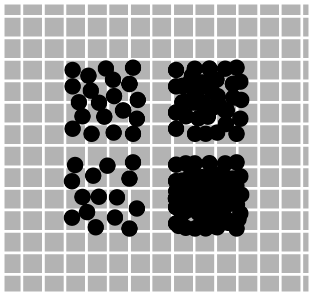

`geom_quantile()`

Continuous Discrete


`geom_bar()`

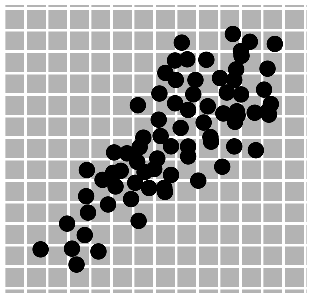

`geom_boxplot()`

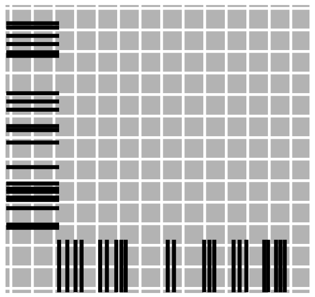

`geom_violin()`

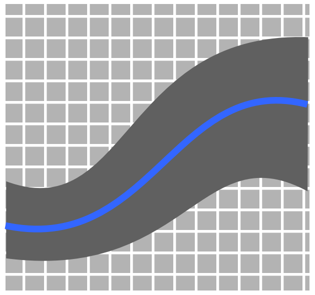

`geom_jitter()`

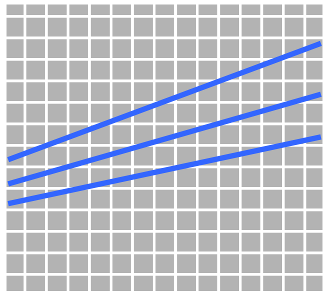
Step 2: Find the function
Plotting two variables

Continuous Continuous

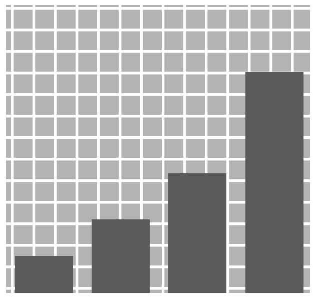

geom_point()

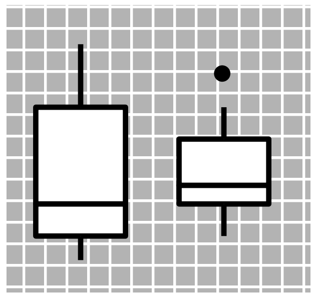

geom_rug()

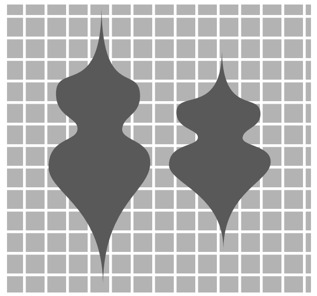

geom_smooth()

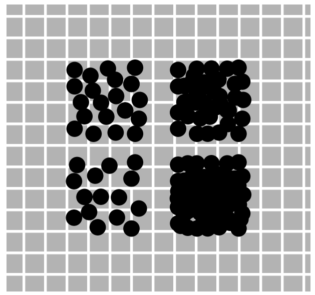

geom_quantile()

Continuous Discrete


geom_bar()


geom_boxplot()


geom_violin()


geom_jitter()

Step 3: Transform data
Some graphs might require transformed data input

- It is quite rare that you can plot your data right away, i.e. certain **plots have requirements** on how the data should look like.
- In most cases it is **necessary to transform** your data before plotting it.
- Examples:

  Lecture 2

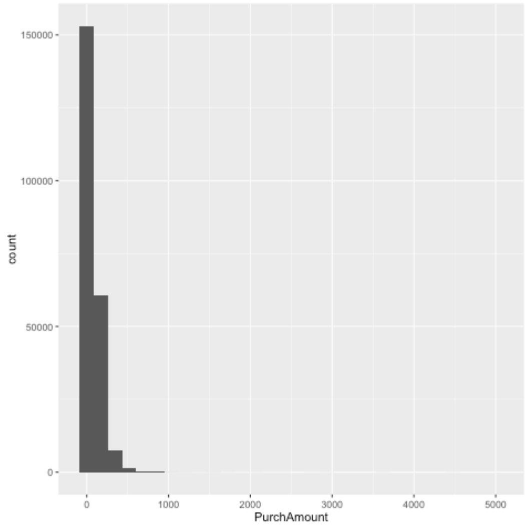
  Lecture 7

  same as last unit

  - Transform times and dates for aggregation of month or years
  - Group data for better overview
  - Logarithmic transformations for nicer distributions

Step 4: Create the plot
Example 1: Create a histogram

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

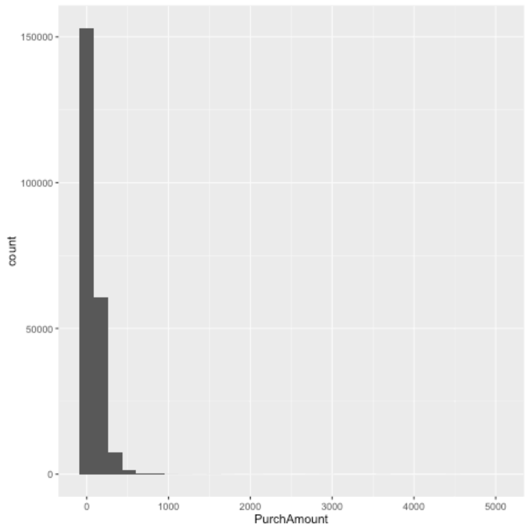


```
ggplot(myData, aes(PurchAmount)) + geom_histogram()
```

Histogram

Step 4: Create the plot
Example 1: Create a histogram

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...



```
ggplot(myData, aes(PurchAmount)) + geom_histogram()
```

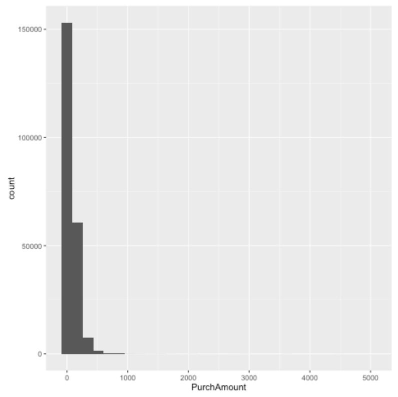
Histogram

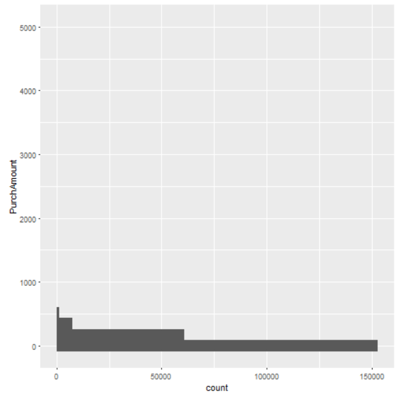
Step 4: Create the plot
Example 1: Flip the coordinates of the histogram

PurchAmount
199.95
199.95
99.95
39.95
79.95
…

+

No plot has been created yet





```
ggplot(myData,
aes(PurchAmount))
```

```
+ geom_histogram()
```

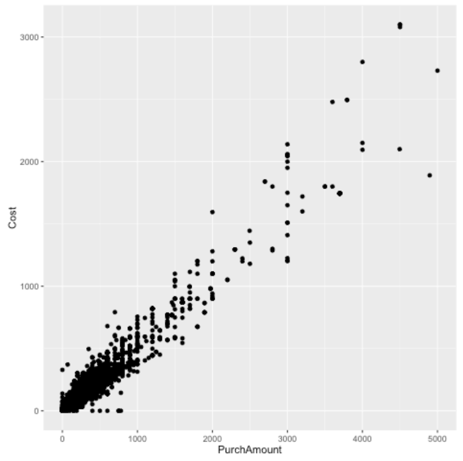
```
+ coord_flip()
```

Note: `coord_cartesian()` is assumed as default [1]

Flips coordinates [2]

Step 4: Create the plot
Example 2: Create a scatterplot

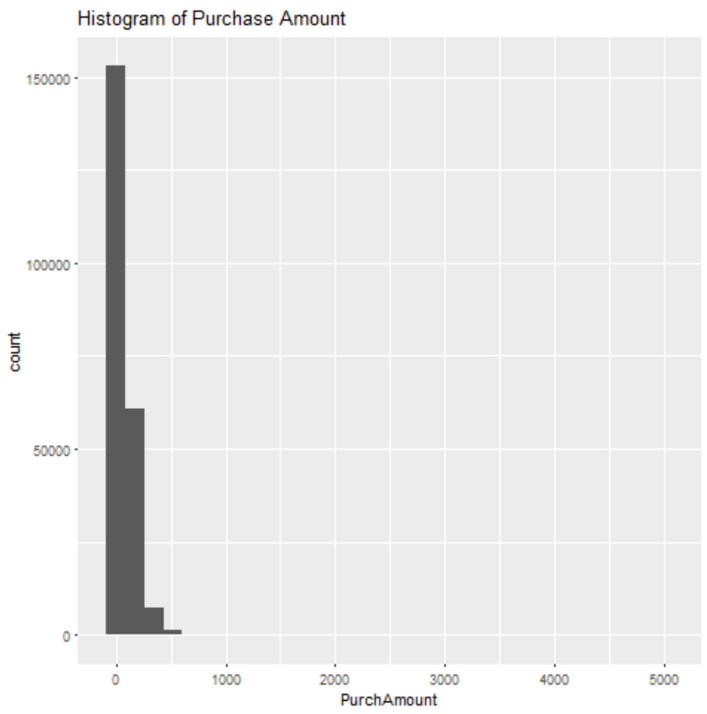
Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

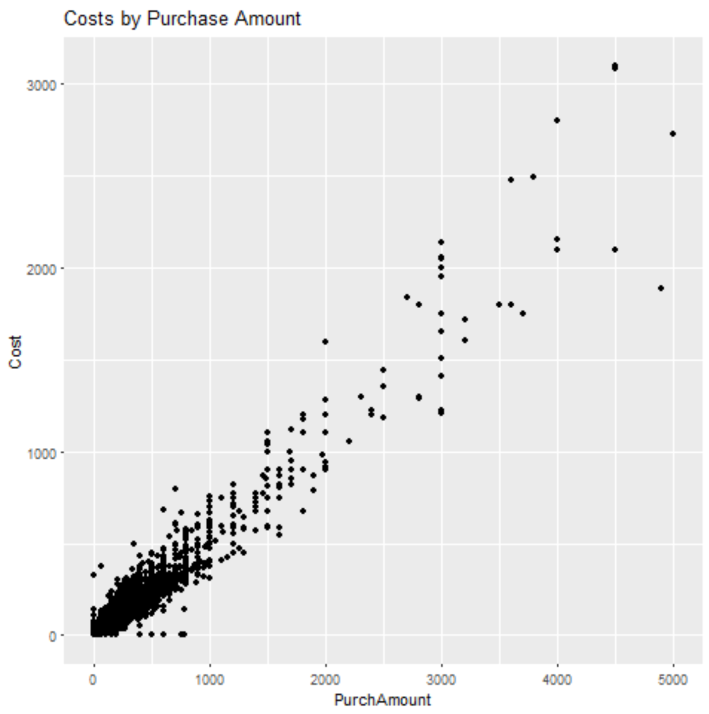


```
ggplot(myData, aes(x=PurchAmount, y =Cost))
                + geom_point()
```

Scatterplot

Step 5: Improve aesthetic features of the plot
Plot title



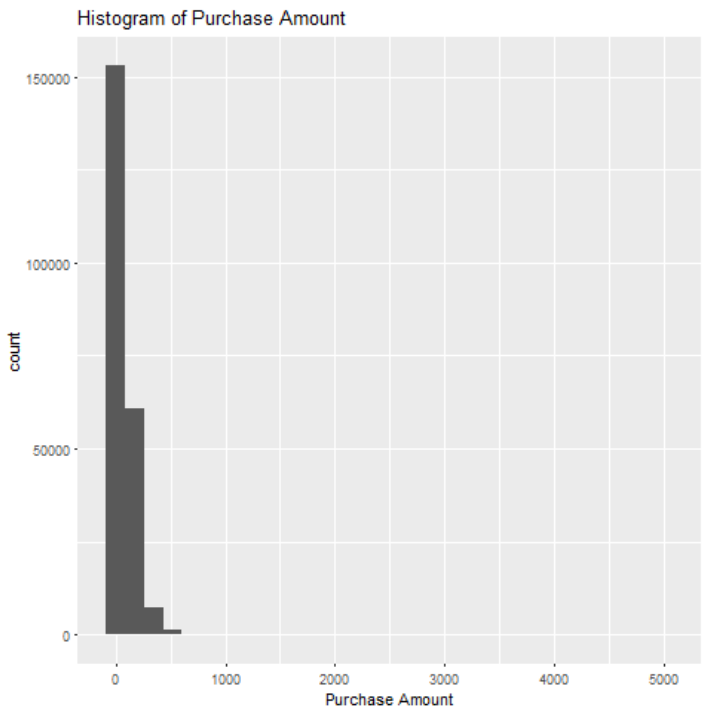


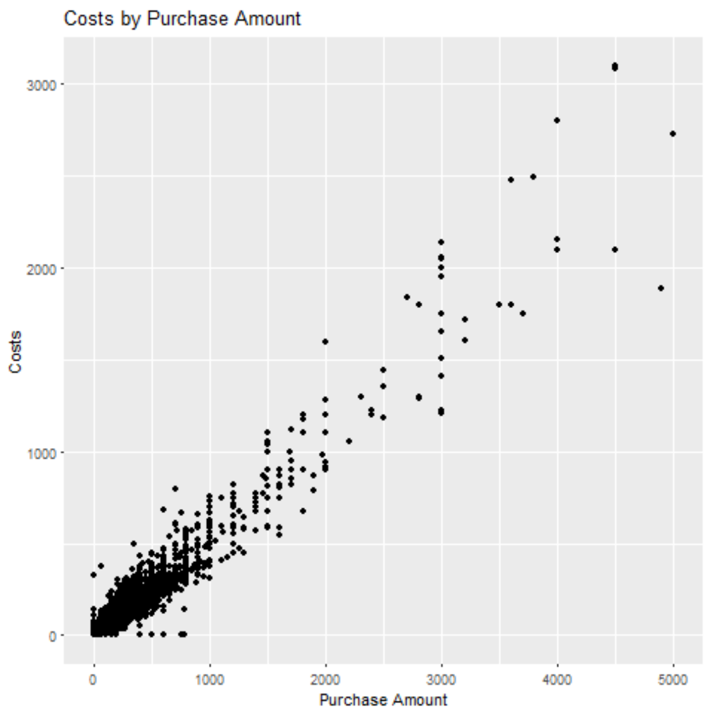
```
ggplot(myData, aes(PurchAmount))
+ geom_histogram() +
 ggtitle("Histogram of Purchase
  Amount")
```

Add title

```
ggplot(myData,aes(x=PurchAmount,
  y =Cost)) + geom_point() +
  ggtitle("Costs by
   Purchase Amount")
```

Step 5: Improve aesthetic features of the plot
Axis labels





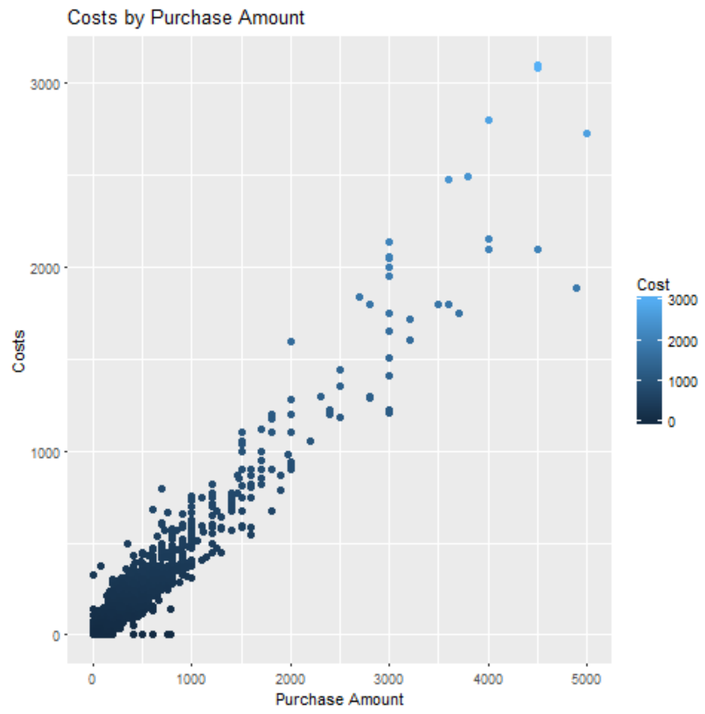
```
ggplot(myData, aes(PurchAmount))+
  geom_histogram()+ … +
  xlab("Purchase Amount")
```

Add x axis label 1

```
ggplot(myData,aes(x=PurchAmount,
  y =Cost)) + geom_point() + … +
  xlab("Purchase Amount") +
  ylab("Costs")
```

Add axes labels 2

Step 5: Improve aesthetic features of the plot
Change point size and color



1 To create faster plots you can also use the `qplot`-package. But it offers less customizations.

2 Specify color. In this case `color = Cost`

```
ggplot(myData, aes(x=PurchAmount, y =Cost, color=Cost)) +
                  geom_point(size=2) + …
```

3 Specify point size