Recall

- Set up R & R studio (rcloud)
- Elementary commands on R

  *, /, +, −
- vectorization as a tool in R
- Data frames in R.
  - in built functions: - c( )
    - mean( )
- Data - { Categorical / numerical

ggplot – GRAMMAR OF GRAPHICS

Coordinate chart

Data frame

1st layer

```
ggplot(data = <DATA>)

+ <GEOM_FUNCTION>(

mapping = aes(<MAPPINGS>),

stat = <STAT>,

position = <POSITION>

) +

<COORDINATE_FUNCTION> +

<FACET_FUNCTION>
```

- `aes()`- Associate the `name of the aesthetic` to the `name of the variable.`
  - the function gathers together each of the aesthetic mappings used by a layer.
  - passes them to the layer's mapping argument.
  - selects a reasonable scale to use with the aesthetic, and it constructs a legend (or axis labels) that explains the mapping between levels and values.
- Above example
  - Name=colour and Variable=class.
  - Scaling: assigns a unique level of the aesthetic `colour` to a unique level to the variable `class`.
  - Other aesthetics include : shape and size.

```
ggplot(data = <DATA>)

+ <GEOM_FUNCTION>(

mapping = aes(<MAPPINGS>),

stat = <STAT>,

position = <POSITION>

) +

<COORDINATE_FUNCTION> +

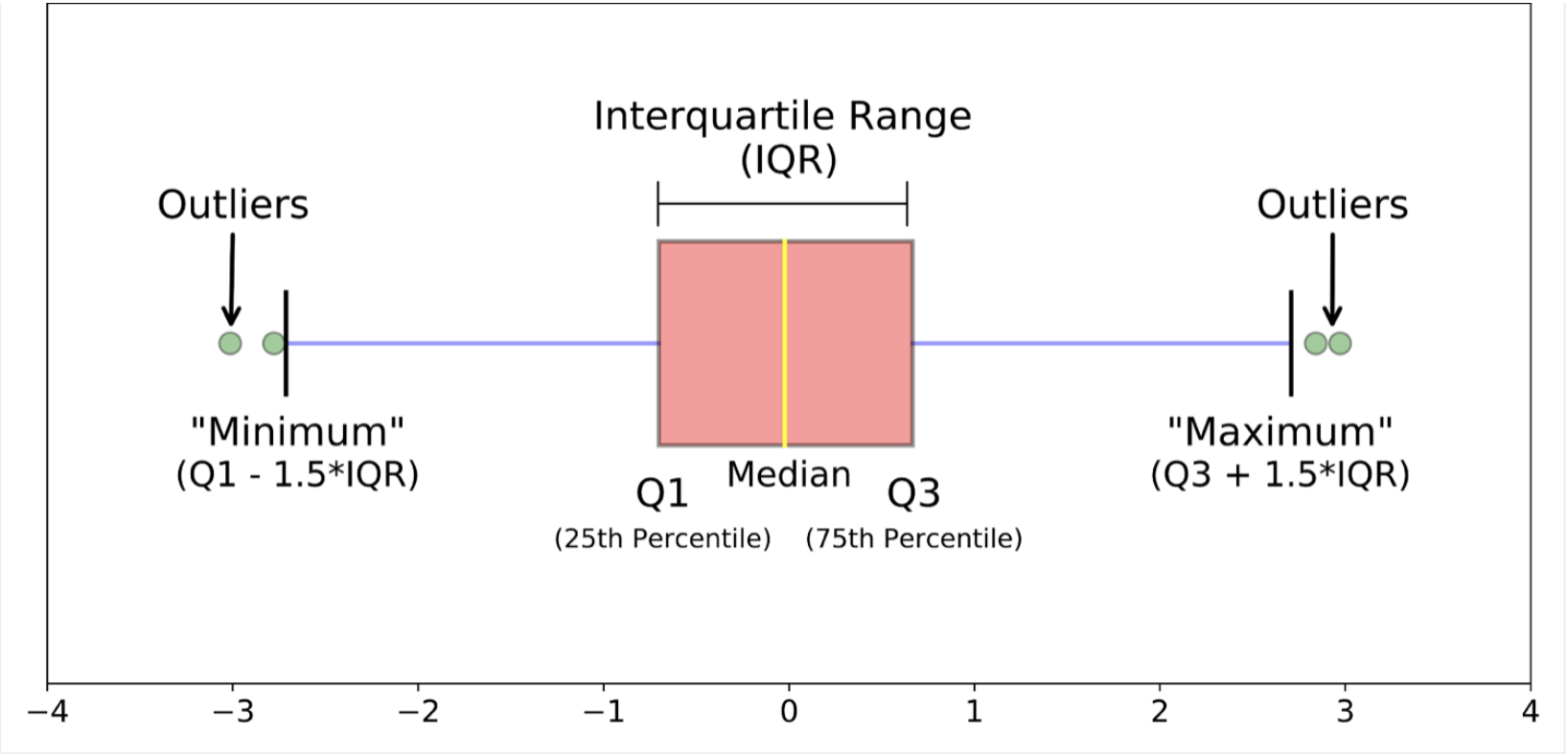
<FACET_FUNCTION>
```

- As `aesthetics` was used to add an additional variable to the plot, another way is to add facets (useful for categorical variables).
- `facet_wrap` splits plot by a single variable into subplots that each display one subset of the data.

Coordinate functions on the chart

Box-Plot

- The boxplot is used to summarize data using the five number summary.
- From the display one can check easily if the data is symmetric or has suspected outliers.
- Its simplicity is its feature.

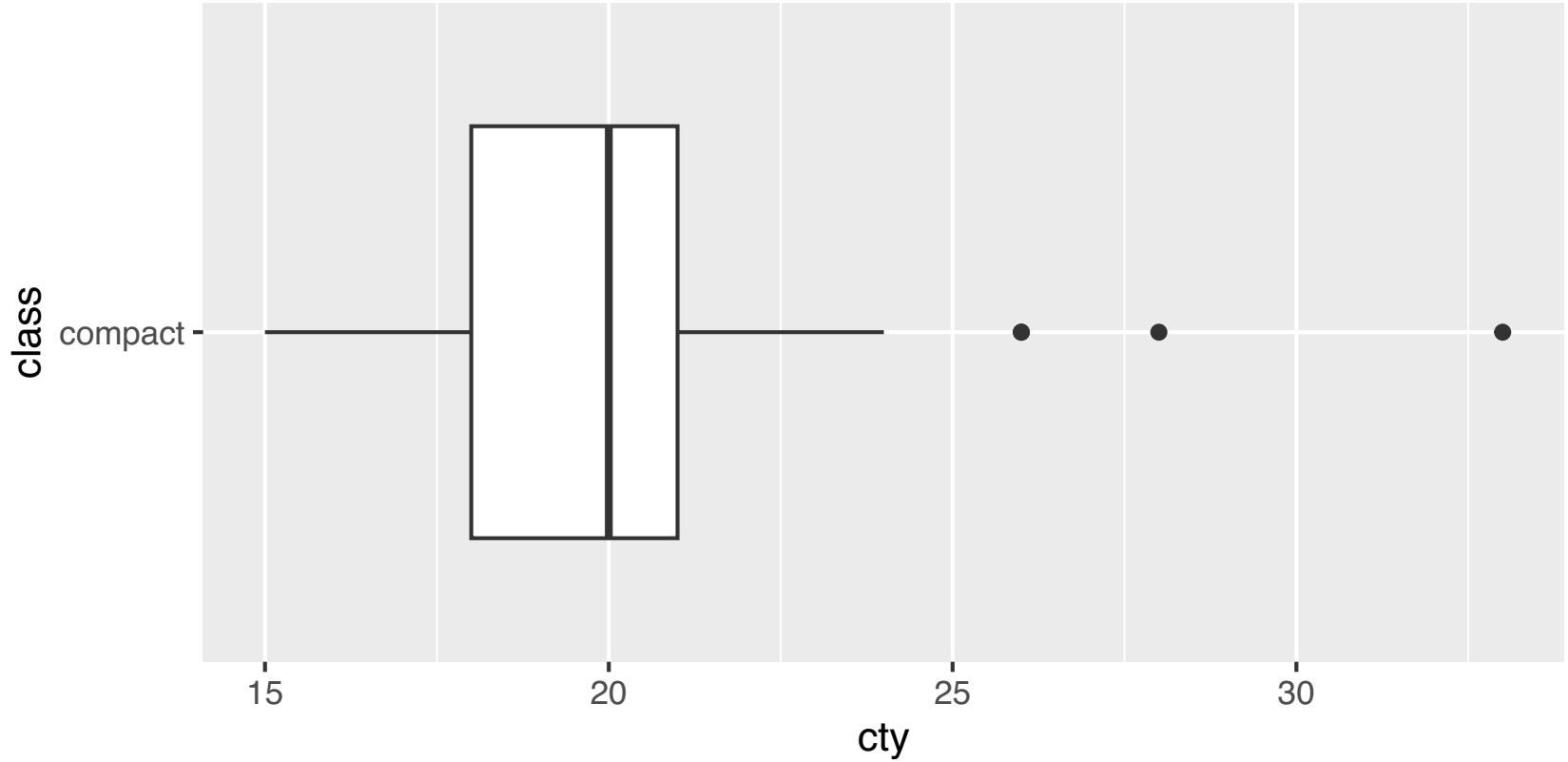


Source: https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51

BoxPlot- cty for compact

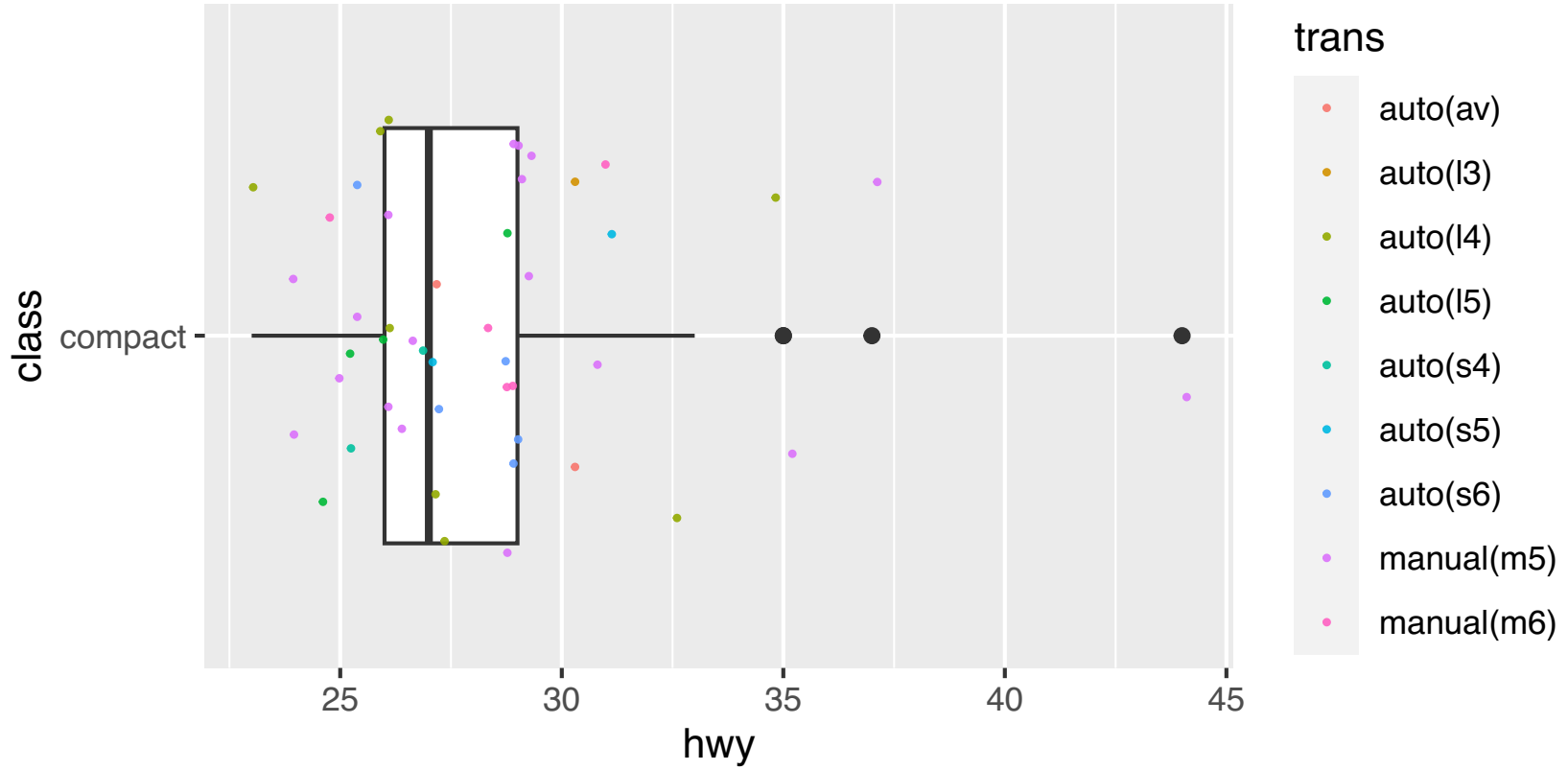
subset: "picks out" all data in "mpg" whose variable class is compact

```
> library(tidyverse)
> mpgcs = subset(mpg, mpg$class=="compact")
> ggplot(data = mpgcs,
+        mapping = aes(x = class, y = cty)) +
+   geom_boxplot()+
+   coord_flip()
```



BoxPlot

```
> ggplot(data = mpgcs, mapping = aes(x = class, y = hwy)) +
+   geom_boxplot() +
+     geom_jitter(aes(color=trans), size=0.4, alpha=0.9)+
+     coord_flip()
```



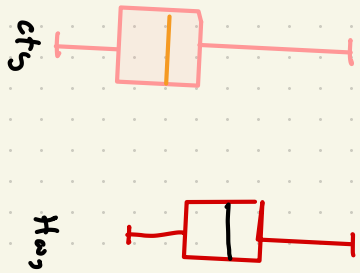
mpg ——— subset —— Compact cars

Box plot

- city miles per gallon
- Hwy miles per gallon — lager with transmission

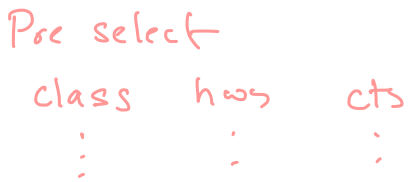
Ex:- Compare using boxplot

Compact - Hwy vs city

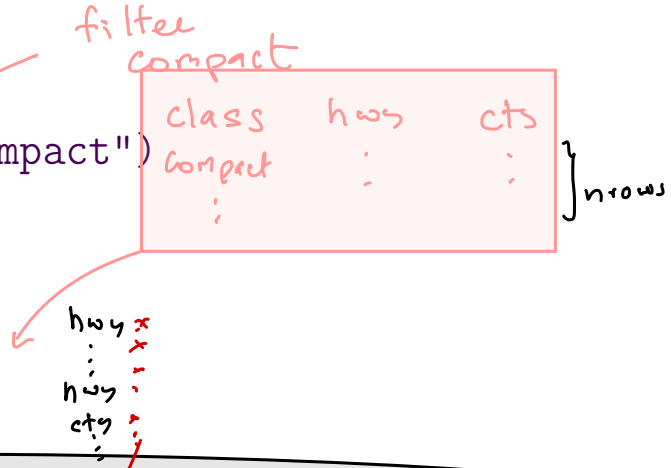


Subset command

```
> library(tidyverse)
> mpgr = mpg[c("class","hwy","cty")]
```



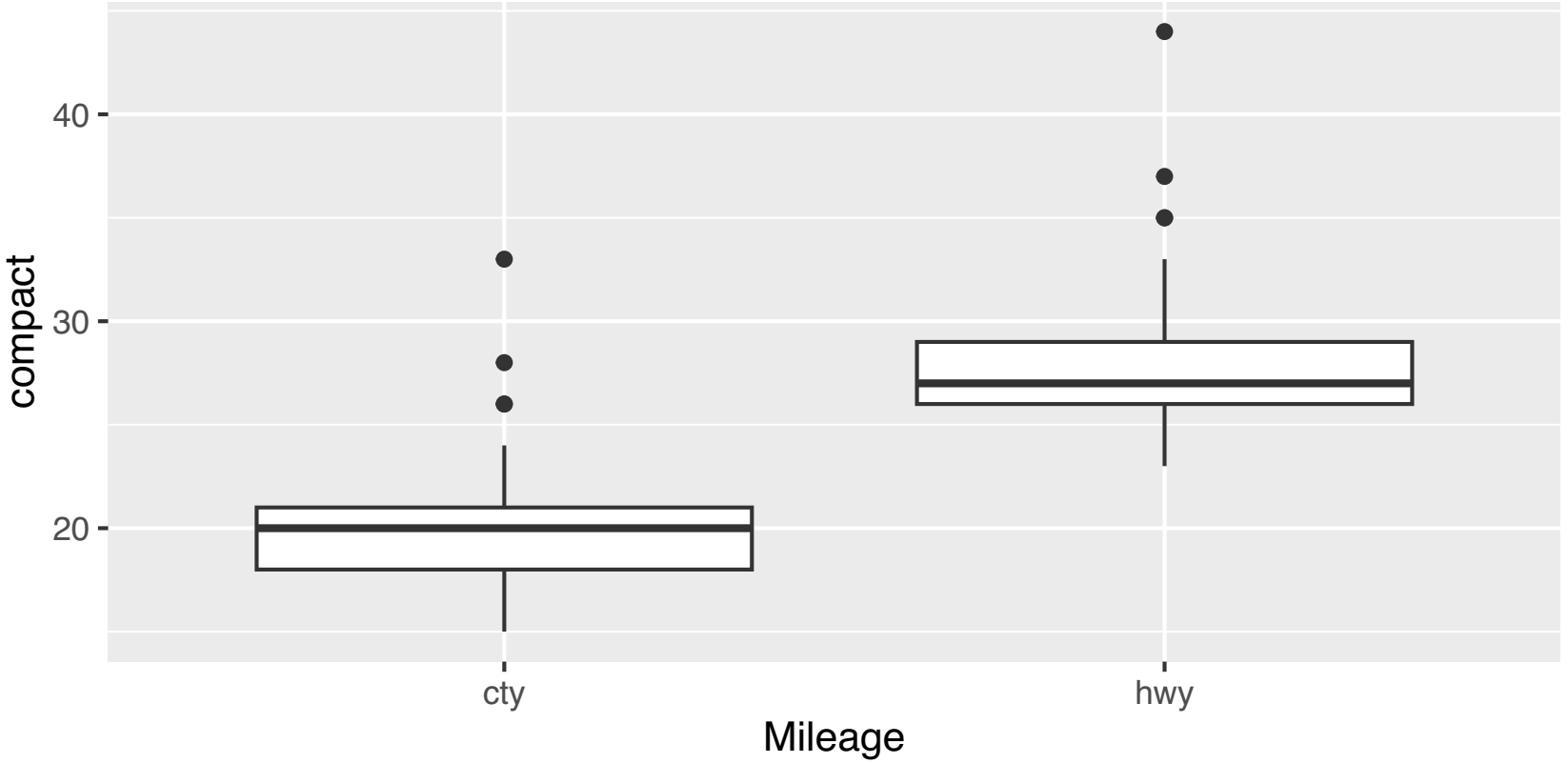

```
> mpgrc = subset(mpgr, mpgr$class=="compact")
```




```
> rdf <- data.frame(
+   Mileage=c( rep("hwy",nrow(mpgrc)), rep("cty",nrow(mpgrc))),
+   compact=c( mpgrc$hwy, mpgrc$cty)
+ )
```

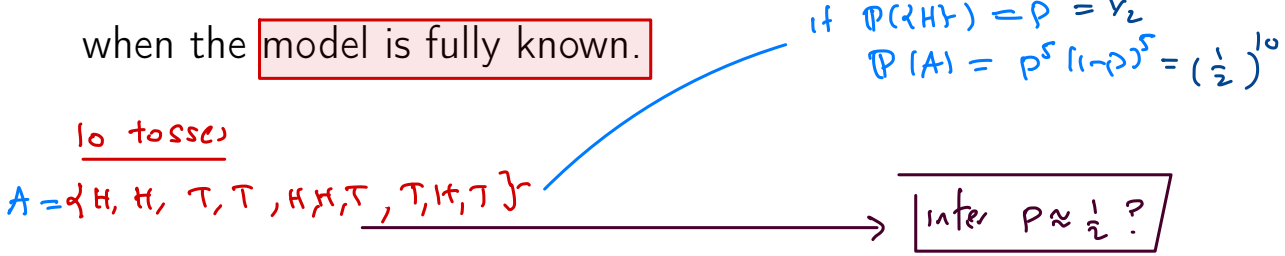
BoxPlot-Comparison Hwy vs cty for models

```
> library(tidyverse)
> mpgr = mpg[c("class","hwy","cty")]
> mpgrc = subset(mpgr, mpgr$class=="compact")
> rdf <- data.frame(
+   Mileage=c( rep("hwy",nrow(mpgrc)), rep("cty",nrow(mpgrc))),
+   compact=c( mpgrc$hwy, mpgrc$cty)
+ )
> ggplot(data = rdf, mapping = aes(x = Mileage, y = compact)) + geom_boxplot()
```



Probability Distributions

- Probability– the study of models for (random) experiments when the model is fully known.



- Statistics– to infer about the unknown aspects of the model based on observed outcomes of the experiment (when the model is not fully known)

- We will now explore how to simulate (random) data from a given distribution.

Generating Random Data

- Historically Random number tables were used to generate random numbers. ?

  ← Worksheet 2

- Presently many alogrithms produce a *pseudo-random number generator*.

- Questions

  - How algorithms work to generate random numbers ?

    Deep subject — Computer Science / Number theory

  - How to generate (random) data from a given distribution ?

    $Y = F_X(X)$?

    Hint :- X – random variable

    $F_X(x) = \mathbb{P}(X \le x) = \sum_{x_i \in \text{Range}(X),\ x_i \le x} \mathbb{P}(X = x_i) = \int_{-\infty}^{x} f(y)\,dy$

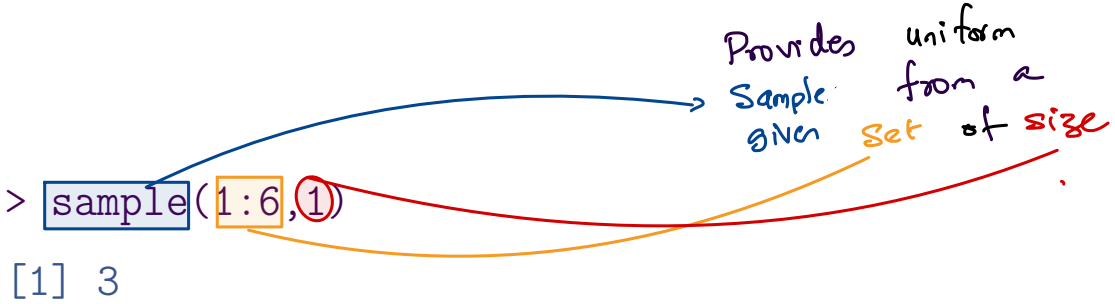
    Q: Find distribution function of Y

  - How to generate using R?

Generating Random data in R

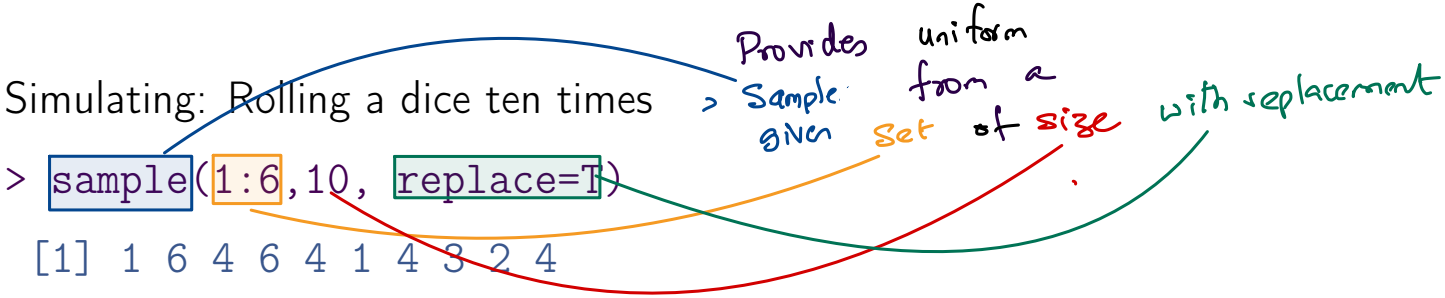
$X \sim \textbf{Uniform}(\{1, 2, \ldots, n\})$: Let $n \geq 1$ be an integer.

If X is a random variable such that
$P(X = k) = \frac{1}{n}$ for all $1 \leq k \leq n$




```
> sample(1:6,1)
[1] 3
```

Simulating: Rolling a dice ten times




```
> sample(1:6,10, replace=T)
 [1] 1 6 4 6 4 1 4 3 2 4
```

Generating Random data in R

$X \sim$ **Bernoulli**(p):

Let $0 < p < 1$

If X is a random variable such that

$P(X = 1) = p = 1 - P(X = 0)$

Simulating: Tossing a fair coin

```
> sample(c(0,1), 1)
[1] 1
```

Simulating: Tossing a biased coin

```
> sample(c(0,1), 1, prob = c(0.3,0.7))
[1] 1
```

Simulating: Tossing a biased coin 10 times

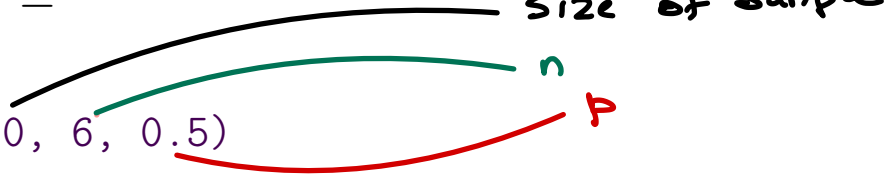
```
> sample(c(0,1), 10, replace =T, prob = c(0.6,0.4))
 [1] 0 1 0 1 0 0 1 0 0 0
```

Generating Random data in R

$X \sim \textbf{Binomial}(n, p)$: Let $0 \leq p \leq 1$ and let $n \geq 1$ be an integer. If X is a random variable taking values in $\{0, 1, \ldots, n\}$ having a probability mass function

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

for all $0 \leq k \leq n$.




```
> rbinom(10, 6, 0.5)
 [1] 3 2 4 2 3 4 1 1 3 3
```

```
> rbinom(n, size,prob)
```

```
> rbinom(10, 30,0.3)
 [1] 11  7 13  7  8 10  6  8 11  8
```

Functions in R

- Created using the `function()` command and are stored as `R` objects of class "function".

- ```
  f <- function(<arguments>) {
        ## Do something interesting
  }
  ```

- can be passed as arguments to other functions

- can be nested, so that you can define a function inside of another function

- The return value of a function is the last expression in the function body to be evaluated.
  ```

Functions in R

```
> f <- function(x,y) {
+         x+y
+ }
> f(2,2)
[1] 4
```

```
> f = function(x){dbinom(x,5,0.3)}
> f(0:5)
[1] 0.16807 0.36015 0.30870 0.13230 0.02835 0.00243
```

Generating Random data in R

```
> rbinom(1000, 10,0.5)
> library(tidyverse)
> df= data.frame(x=rbinom(1000, 10, 0.5))
> f = function(x){dbinom(x,10,0.5)}
> df2 = data.frame(x=0:10, y=f(0:10))
> ggplot()+
+ geom_histogram(df, mapping=aes(x = x, y =..density.., fill=..x..),breaks=0:10)+
+ scale_fill_viridis()+ geom_line(df2, mapping=aes(x = x,y =y))
```

