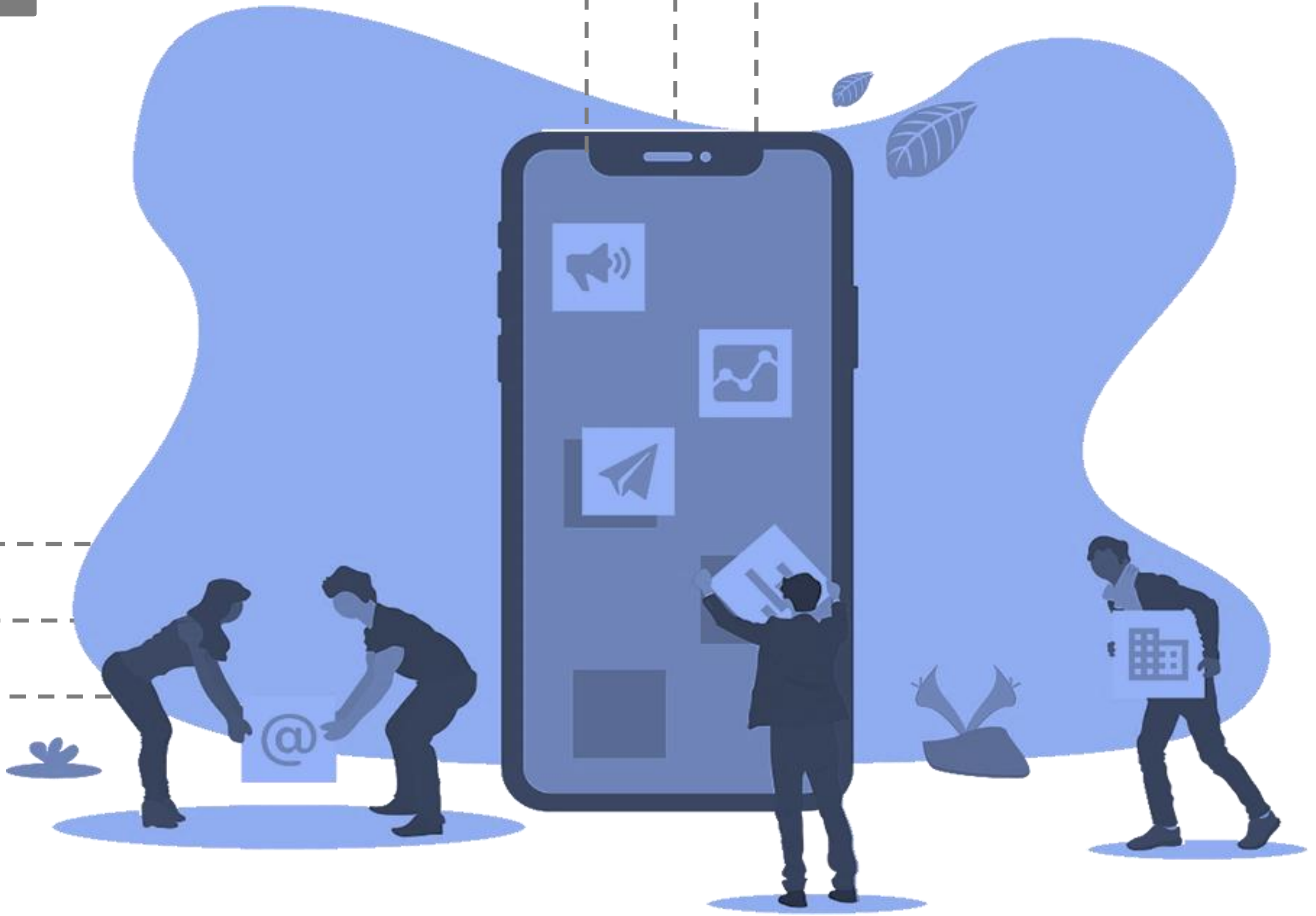
The Data Science Track



Prepared By: R. Daynolo

INTRODUCTION TO DATA SCIENCE WITH R

17. SIMULATION

GENERATING RANDOM NUMBERS

Functions for probability distributions in R

- `rnorm`: generate random Normal variates with a given mean and standard deviation

- `dnorm`: evaluate the Normal probability density (with a given mean/SD) at a point (or vector of points)

- `pnorm`: evaluate the cumulative distribution function for a Normal distribution

- `rpois`: generate random Poisson variates with a given rate

GENERATING RANDOM NUMBERS

Probability distribution functions usually have four functions associated with them. The functions are prefixed with a

- `d` for density
- `r` for random number generation
- `p` for cumulative distribution
- `q` for quantile function

GENERATING RANDOM NUMBERS

Working with the Normal distributions requires using these four functions

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```

If Φ is the cumulative distribution function for a standard Normal distribution, then `pnorm(q)` $= \Phi(q)$ and `qnorm(p)` $= \Phi^{-1}(p)$.

GENERATING RANDOM NUMBERS

```
> x <- rnorm(10)
> x
 [1]  1.2495116  0.1417660  1.2225387  1.6392652 -0.1349005
 [6] -0.1308487 -1.2000805  1.8147253 -0.7597239  0.3247771
> x <- rnorm(10, 20, 2)
> x
 [1] 18.24445 17.17698 19.14051 22.42392 15.88907 19.52422
 [7] 21.69810 15.53143 19.98573 19.30795
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  15.53   17.44   19.22   18.89   19.87   22.42 
```

ANALYSIS STRUCTURE ALGORITHM PROCESS PROGRAMMING SOLVING KNOWLEDGE

GENERATING RANDOM NUMBERS

Setting the random number seed with `set.seed` ensures reproducibility.

```
> set.seed(1)
> rnorm(5)
[1] -0.6264538  0.1836433 -0.8356286  1.5952808  0.3295078
> rnorm(5)
[1] -0.8204684  0.4874291  0.7383247  0.5757814 -0.3053884
> set.seed(1)
> rnorm(5)
[1] -0.6264538  0.1836433 -0.8356286  1.5952808  0.3295078
```

Note: Always set the random number seed when conducting a simulation!

Generating Poisson data

```
> rpois(10, 1)
 [1] 0 0 1 1 2 1 1 4 1 2
> rpois(10, 2)
 [1] 4 1 2 0 1 1 0 1 4 1
> rpois(10, 20)
 [1] 19 19 24 23 22 24 23 20 11 22
> ppois(2, 2) ##Cumulative distribution
[1] 0.6766764   ##P(x≤2)
> ppois(4, 2)
[1] 0.947347    ##P(x≤4)
> ppois(6, 2)
[1] 0.9954662   ##P(x≤6)
```

GENERATING RANDOM NUMBERS FROM A LINEAR MODEL

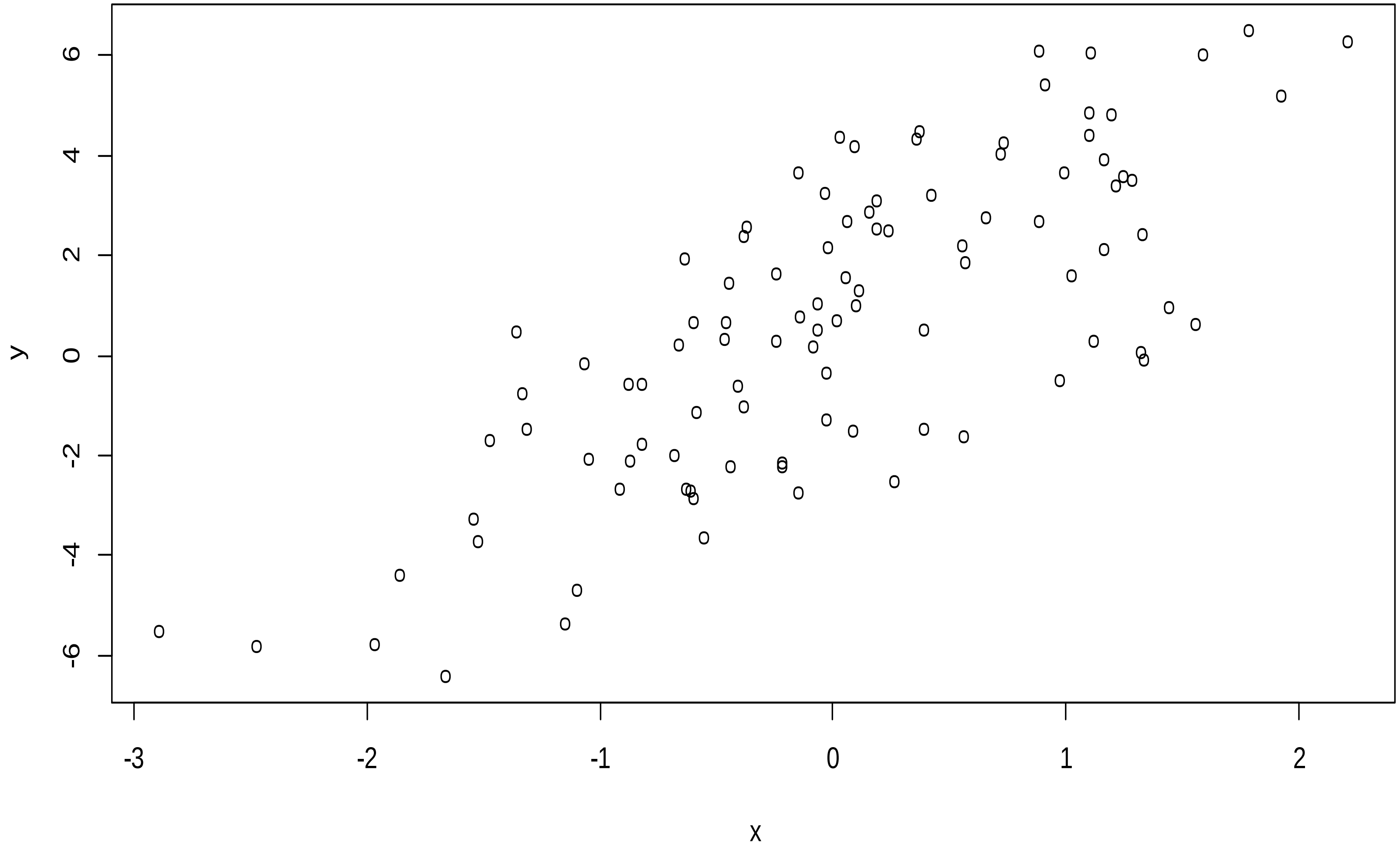
Suppose we want to simulate from the following linear model

$$y = \beta_0 + \beta_1 + \varepsilon$$

where $\varepsilon \sim N(0, 2^2)$. Assume $x \sim N(0, 1^2)$, $\beta_0 = 0.5$ and $\beta_1 = 2$.

```
> set.seed(20)
> x <- rnorm(100)
> e <- rnorm(100, 0, 2)
> y <- 0.5 + 2 * x + e
> summary(y)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-6.4084 -1.5402  0.6789  0.6893  2.9303  6.5052
> plot(x, y)
```

GENERATING RANDOM NUMBERS FROM A LINEAR MODEL

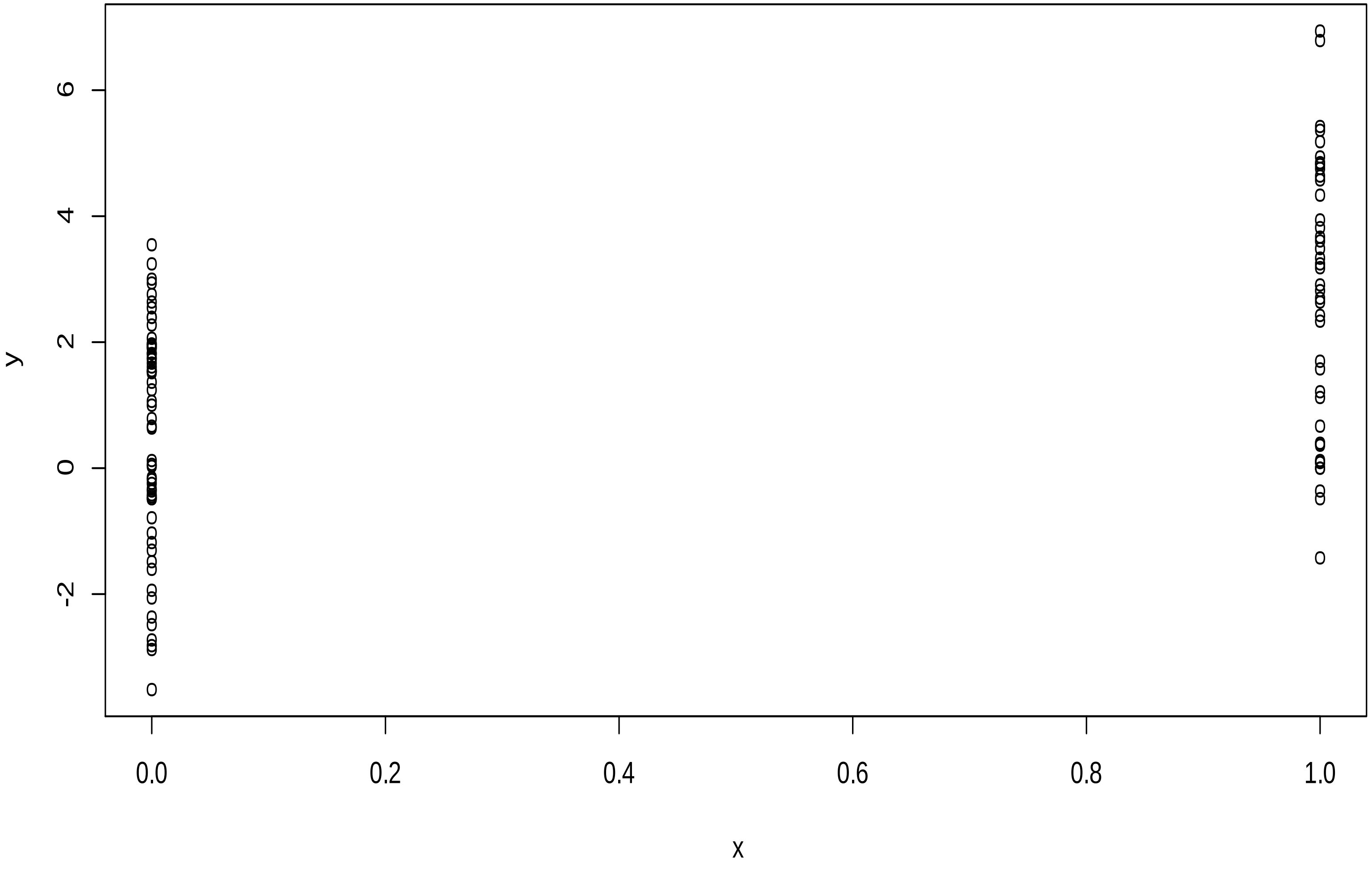


GENERATING RANDOM NUMBERS FROM A LINEAR MODEL

What if x is binary?

```
> set.seed(10)
> x <- rbinom(100, 1, 0.5)
> e <- rnorm(100, 0, 2)
> y <- 0.5 + 2 * x + e
> summary(y)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
-3.4936 -0.1409  1.5767  1.4322  2.8397  6.9410 
> plot(x, y)
```

GENERATING RANDOM NUMBERS FROM A LINEAR MODEL



ANALYSIS STRUCTURE ALGORITHM PROCESS PROGRAMMING SOLVING KNOWLEDGE

GENERATING RANDOM NUMBERS FROM A GENERALIZED LINEAR MODEL

Suppose we want to simulate from a Poisson model where

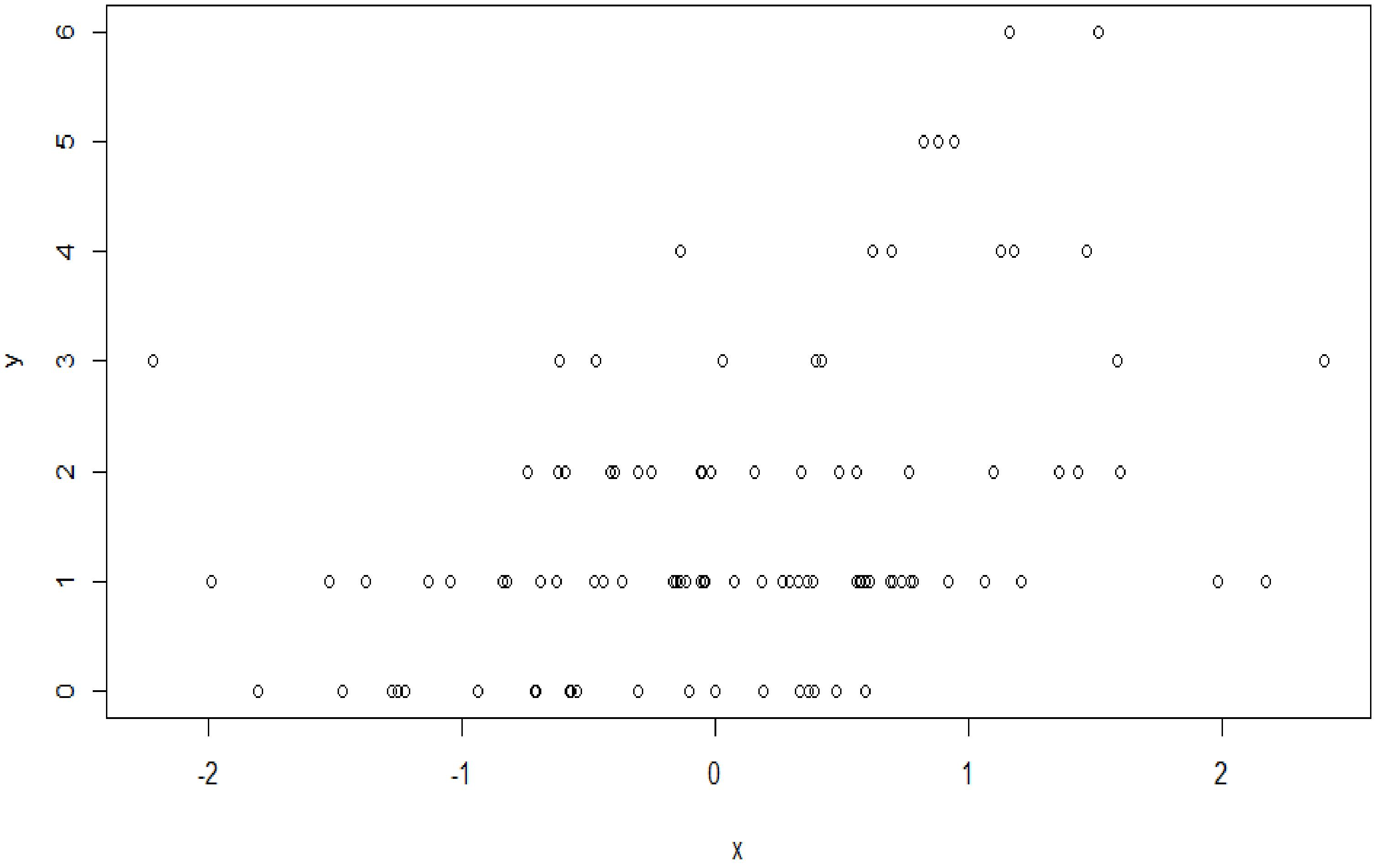
$Y \sim Po\ (\mu)$

$log\ (\mu) =\ \beta_0 + \beta_1 x$ and $\beta_0 = 0.5$ and $\beta_1 = 0.3$. We need `rpois` function

```
> set.seed(1)
> x <- rnorm(100)
> log.mu <- 0.5 + 0.3 * x
> y <- rpois(100, exp(log.mu))
> summary(y)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    1.00    1.00    1.55    2.00    6.00
> plot(x, y)
```

ANALYSIS STRUCTURE ALGORITHM PROCESS PROGRAMMING SOLVING KNOWLEDGE

GENERATING RANDOM NUMBERS FROM A GENERALIZED LINEAR MODEL



RANDOM SAMPLING

The `sample` function draws randomly from a specified set of (scalar) objects allowing you to sample from arbitrary distributions.

```
> set.seed(1)
> sample(1:10, 4)
[1] 3 4 5 7
> sample(1:10, 4)
[1] 3 9 8 5
> sample(letters, 5)
[1] "q" "b" "e" "x" "p"
> sample(1:10) ## permutation
 [1]  4  7 10  6  9  2  8  3  1  5
> sample(1:10)
 [1]  2  3  4  1  9  5 10  8  6  7
> sample(1:10, replace = TRUE)
 [1] 2 9 7 8 2 8 5 9 7 8
```

SIMULATION

Summary

- Drawing samples from specific probability distributions can be done with `r*` functions
- Standard distributions are built in: Normal, Poisson, Binomial, Exponential, Gamma, etc.
- The `sample` function can be used to draw random samples from arbitrary vectors
- Setting the random number generator seed via `set.seed` is critical for reproducibility