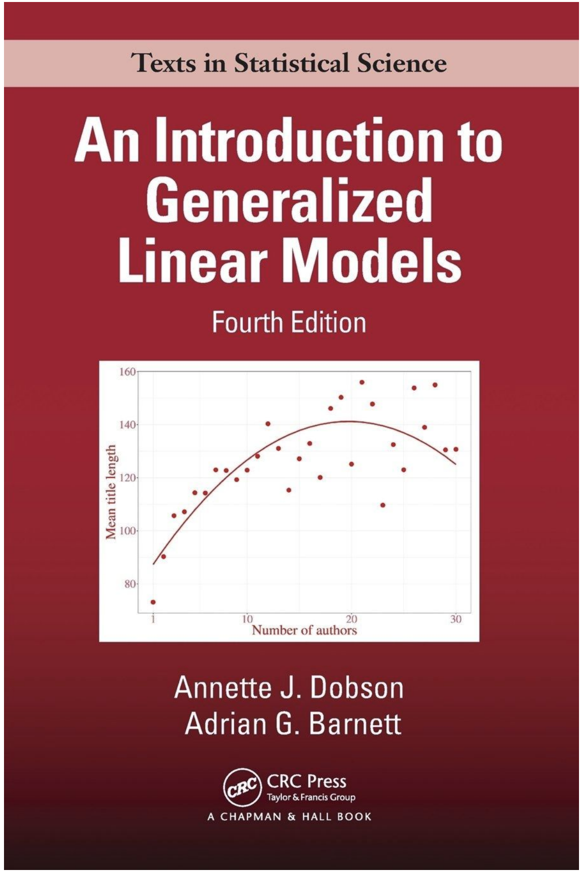
Introduction to Data Science

Distributions

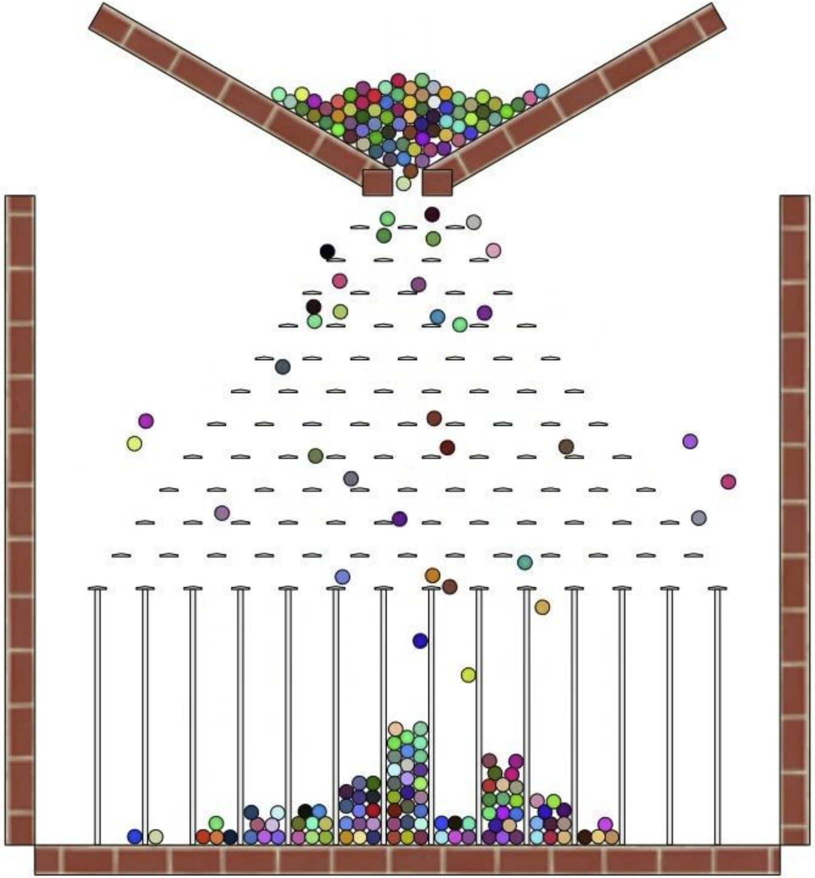
Prof. Dr. Ralf Lämmel & M.Sc. **Johannes Härtel**
(johanneshaertel@uni-koblenz.de)

A book recommendation for this lecture: Especially the first part of this book provides a **formal and concise** introduction to distributions.



Some distributions in real life

Binomial Distribution (special case of Bernoulli distribution if just 0 or 1)



Source

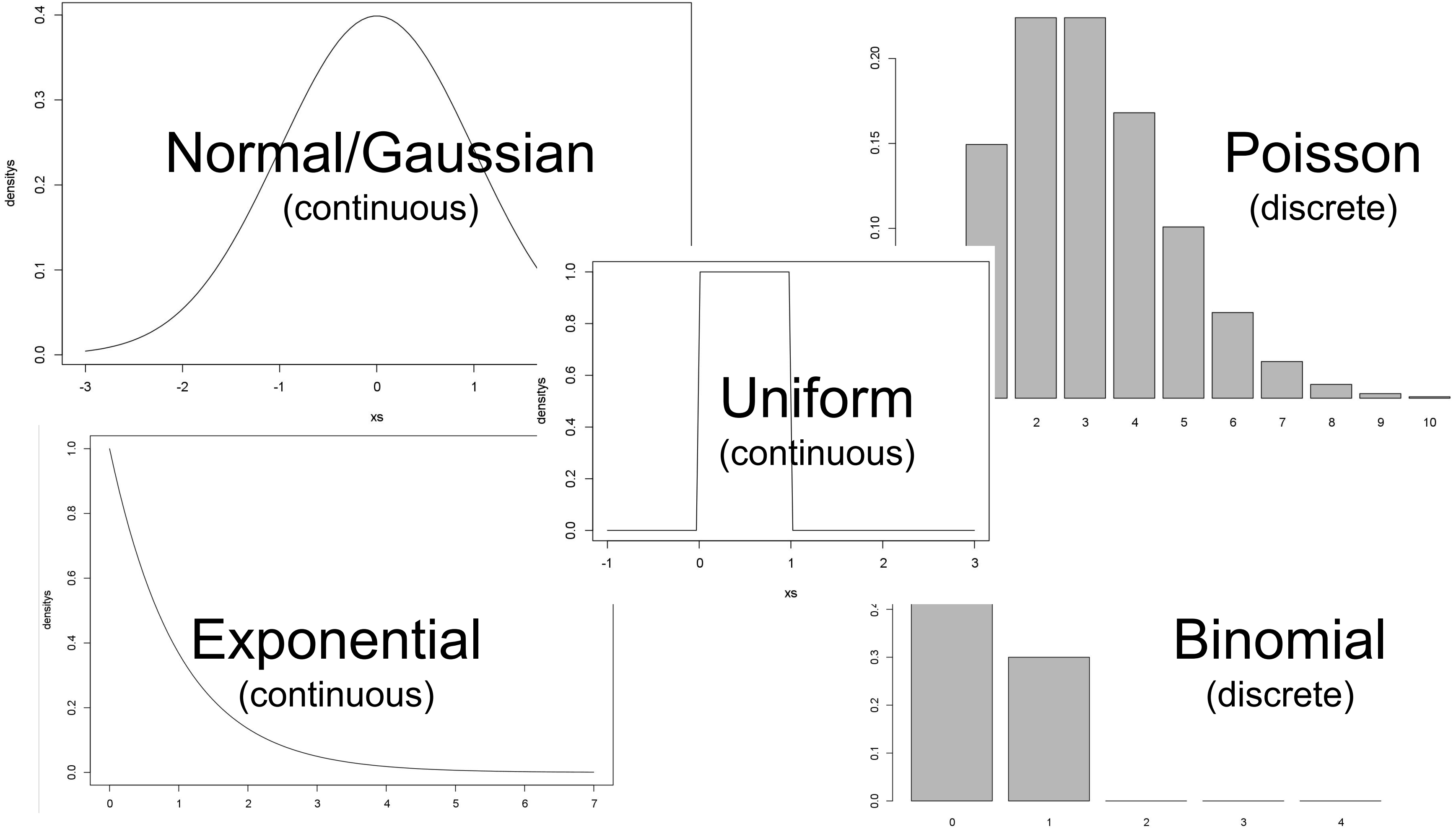
A (discretized) normal distribution

Filter messages by: Sender Recipi

Subject	Correspondents	Date
[Mitarbeiter] Sparkasse, Volksbank, etc.: Phishing-Versuche herausfiltern	Christoph Litauer via Mitarbeiter	10/25/2021, 2:44 PM
[Mitarbeiter] neue Spam-Phishingwelle, Hinweise	Konstantin Root via Mitarbeiter	8/17/2021, 1:31 PM
Re: [Mitarbeiter] Spam, Phishing und Konfiguration der Mailinglisten	Christoph Litauer via Mitarbeiter	4/11/2021, 11:10 AM
[Mitarbeiter] Spam, Phishing und Konfiguration der Mailinglisten	Christoph Litauer via Mitarbeiter	4/9/2021, 1:38 PM
[Mitarbeiter] Phishing-Mail (?)	Christoph Litauer via Mitarbeiter	3/3/2021, 10:53 AM
Achtung vor Phishing: So prüfen Sie die Echtheit einer STRATO E-Mail	STRATO	2/22/2021, 3:35 PM
[Mitarbeiter] Achtung: Sogo-Phishing Mail im Umlauf	Christoph Litauer via Mitarbeiter	5/28/2020, 11:46 AM
[Mitarbeiter] Hochwertige Spear-Phishing-Angriffe gegen deutsche Universitäten und Forschungseinrichtungen	Christian Schneider via Mitarbeiter	12/4/2019, 11:38 AM
[Mitarbeiter] Phishing und die Folgen	Christoph Litauer via Mitarbeiter	6/4/2019, 4:40 PM
[Mitarbeiter] Vorsicht Phishing Mail / Das gleiche wie das letzte mal!!!!!		6/1/2019, 8:32 PM
[Mitarbeiter] Vorsicht Phishing Mail		5/29/2019, 5:10 PM
[Mitarbeiter] Wieder mal ein PHISHING Mail		1/3/2019, 10:52 AM
[Mitarbeiter] Phishing/Virus-Welle		10/6/2017, 9:18 AM
[Studierende] Phishing erkennen und vorbeugen	Christoph Litauer	2/10/2016, 1:11 PM
[Studierende] Achtung Phishing!	Christoph Litauer	7/18/2014, 10:41 AM
[Studierende] Neue Phishing Mails	Uwe Arndt	7/9/2013, 4:05 PM

Number of “Phishing mail” mails in each year.

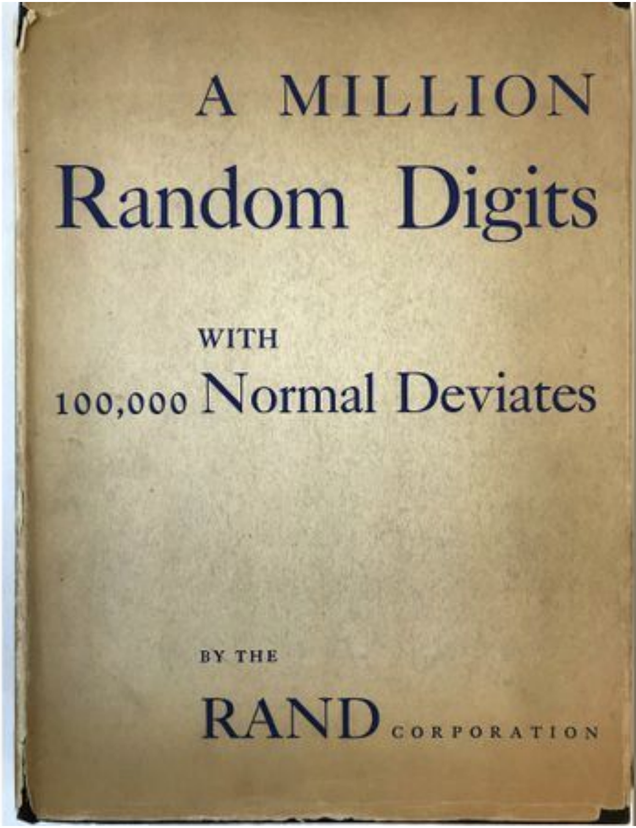
A Poisson distribution


Normal/Gaussian
(continuous)
Poisson
(discrete)
Uniform
(continuous)
Exponential
(continuous)
Binomial
(discrete)
densitys
xs

Ingredients of Distributions

- **Type**: normal, exponential, uniform, Poisson, Binomial (discrete/ continuous) ...
- **Parameters**: mean, standard deviation, lambda, rate, prob, min max …
- **Functions**: (probability) density/mass function (PDF), cumulative distribution function (CDF), quantile function (inverse CDF) and random generation function.

The **random generation** function



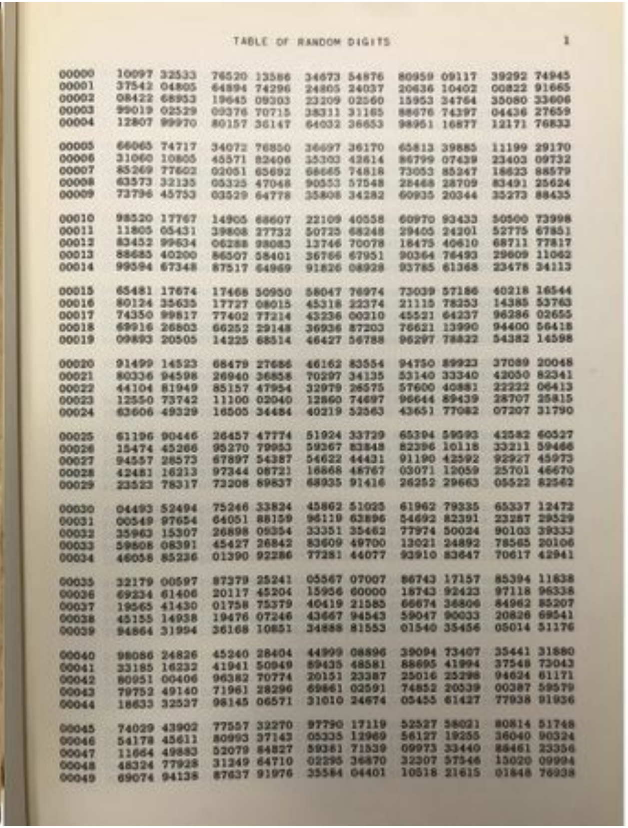


TABLE OF RANDOM DIGITS 1

00000	10097	32533	76520	13586	34673	54876	80959	09117	39292	74945
00001	37542	04805	64894	74296	24805	24037	20636	10402	00822	91665
00002	08422	68953	19645	09303	23209	02560	15953	34764	35080	33606
00003	99019	02529	09376	70715	38311	31165	88676	74397	04436	27659
00004	12807	99970	80157	36147	64032	36653	98951	16877	12171	76833
00005	66065	74717	34072	76850	36697	36170	65813	39885	11199	29170
00006	31060	10805	45571	82406	35303	42614	86799	07439	23403	09732
00007	85269	77602	02051	65692	68665	74818	73053	85247	18623	88579
00008	63573	32135	05325	47048	90553	57548	28468	28709	83491	25624
00009	73796	45753	03529	64778	35808	34282	60935	20344	35273	88435
00010	98520	17767	14905	68607	22109	40558	60970	93433	50500	73998
00011	11805	05431	39808	27732	50725	68248	29405	24201	52775	67851
00012	83452	99634	06288	98083	13746	70078	18475	40610	68711	77817
00013	88685	40200	86507	58401	36766	67951	90364	76493	29609	11062
00014	99594	67348	87517	64969	91826	08928	93785	61368	23478	34113
00015	65481	17674	17468	50950	58047	76974	73039	57186	40218	16544
00016	80124	35635	17727	08015	45318	22374	21115	78253	14385	53763
00017	74350	99817	77402	77214	43236	00210	45521	64237	96286	02655
00018	69916	26803	66252	29148	36936	87203	76621	13990	94400	56418
00019	09893	20505	14225	68514	46427	56788	96297	78822	54382	14598
00020	91499	14523	68479	27686	46162	83554	94750	89923	37089	20048
00021	80336	94598	26940	36858	70297	34135	53140	33340	42050	82341
00022	44104	81949	85157	47954	32979	26575	57600	40881	22222	06413
00023	12550	73742	11100	02040	12860	74697	96644	89439	28707	25815
00024	63606	49329	16505	34484	40219	52563	43651	77082	07207	31790
00025	61196	90446	26457	47774	51924	33729	65394	59593	42582	60527
00026	15474	45266	95270	79953	59367	83848	82396	10118	33211	59466
00027	94557	28573	67897	54387	54622	44431	91190	42592	92927	45973
00028	42481	16213	97344	08721	16868	48767	03071	12059	25701	46670
00029	23523	78317	73208	89837	68935	91416	26252	29663	05522	82562
00030	04493	52494	75246	33824	45862	51025	61962	79335	65337	12472
00031	00549	97654	64051	88159	96119	63896	54692	82391	23287	29529
00032	35963	15307	26898	09354	33351	35462	77974	50024	90103	39333
00033	59808	08391	45427	26842	83609	49700	13021	24892	78565	20106
00034	46058	85236	01390	92286	77281	44077	93910	83647	70617	42941
00035	32179	00597	87379	25241	05567	07007	86743	17157	85394	11838
00036	69234	61406	20117	45204	15956	60000	18743	92423	97118	96338
00037	19565	41430	01758	75379	40419	21585	66674	36806	84962	85207
00038	45155	14938	19476	07246	43667	94543	59047	90033	20826	69541
00039	94864	31994	36168	10851	34888	81553	01540	35456	05014	51176
00040	98086	24826	45240	28404	44999	08896	39094	73407	35441	31880
00041	33185	16232	41941	50949	89435	48581	88695	41994	37548	73043
00042	80951	00406	96382	70774	20151	23387	25016	25298	94624	61171
00043	79752	49140	71961	28296	69861	02591	74852	20539	00387	59579
00044	18633	32537	98145	06571	31010	24674	05455	61427	77938	91936
00045	74029	43902	77557	32270	97790	17119	52527	58021	80814	51748
00046	54178	45611	80993	37143	05335	12969	56127	19255	36040	90324
00047	11664	49883	52079	84827	59381	71539	09973	33440	88461	23356
00048	48324	77928	31249	64710	02295	36870	32307	57546	15020	09994
00049	69074	94138	87637	91976	35584	04401	10518	21615	01848	76938

Some early work on random values.

We want to generate random values (efficiently).

Management Data Base Stystems — R. Benjamin, Editor

Computer Methods for Sampling from the Exponential and Normal Distributions

J. H. Ahrens
Nova Scotia Technical College
and
U. Dieter
Universität Karlsruhe

We see that is might also be called ‘*sampling from a distribution*’.

You will find publications on **efficient method**. For us, it is sufficient to know that such method exist.

A simple random generation function

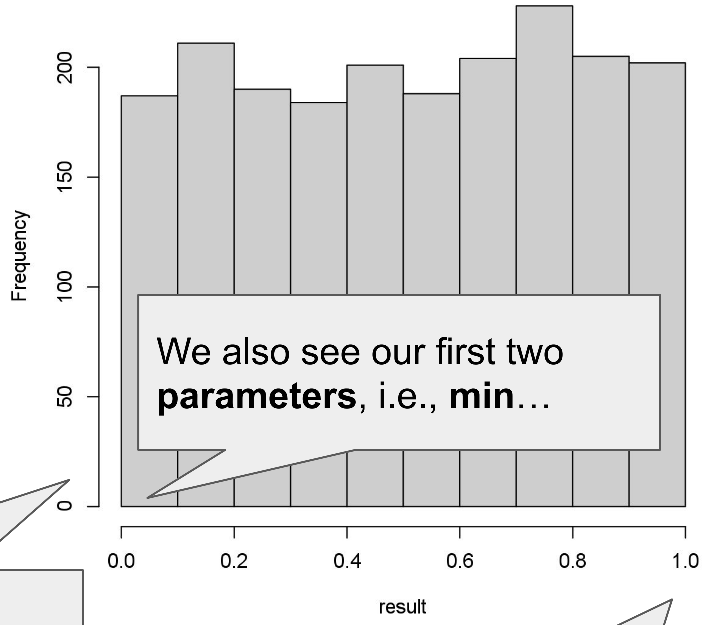
This code shows a Linear congruential generator (Link)

```r
n <- 2000

# I borrowed these constants from a paper.
m <- 2^32
a <- 1103515245
c <- 12345

# d starts with some sort of seed.
d <- 1000
results <- vector(length = n)
for (i in 1:n) {
 d <- (a * d + c) %% m
 results[i] <- d / m
}
```

Plotting the results as histogram

```
 [1] 0.932167650 0.121204376 0.920264259
 [8] 0.775016561 0.357110500 0.270018101
[15] 0.815303341 0.601030588 0.734871149
[22] 0.916099012 0.558927298 0.739606619   ...
[29] 0.368894354 0.891547620 0.640638113
[36] 0.144315481 0.578168631 0.047849417
[43] 0.577669621 0.810473919 0.197372913
                    ⋮
```

The generated random values



We also see our first two **parameters**, i.e., **min**…

We have been sampling from a **uniform distribution**.

… and **max**.

Modifying the parameters

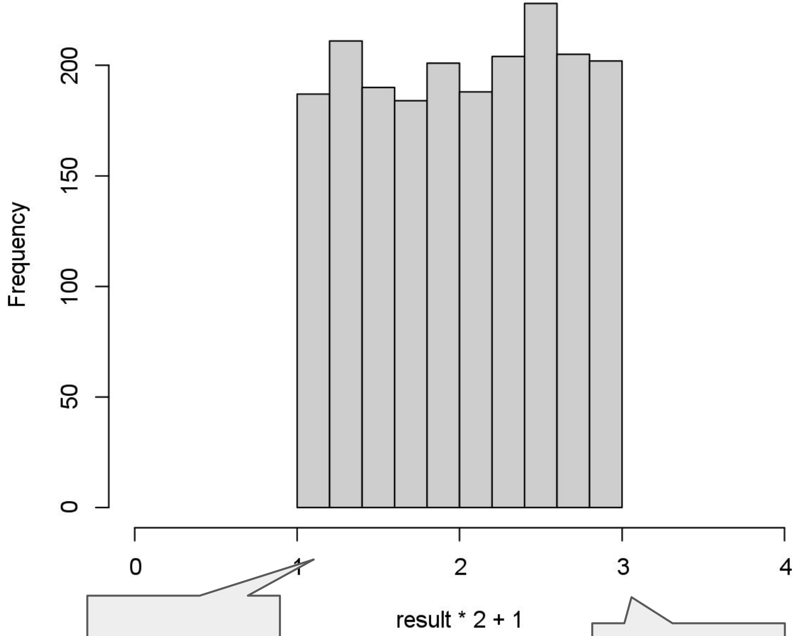
In most case, we want to have different parameters.

```
hist(result * 2 + 1, xlim = c(0, 4))
```

Plotting histogram in R

Modification

Limits for plot x-axis.



new **min**

new **max**

R’s API

We have comparable packages in python, but R is convenient here (link).

Bars are added to keep the tension!

Uniform {stats} R Documentation

The Uniform Distribution

Description

These functions provide information about the uniform distribution on the interval from `min` to `max`. [redacted] and `runif` generates random deviates.

Usage

[redacted]

Here it is called ‘*generates random deviates*’ (╯°□°)╯︵ ┻━┻

```
runif(n, min = 0, max = 1)
```

Changeling the type of a distribution

We can convert a uniform to a normal distribution.

```
results <- NULL
for (x in 1:1000) {
 parts <- runif(n = 1000, min = 0, max = 1)
 results <- c(results, sum(parts))
}

# Some normalization to center on 0 and make standard deviation (sd) equals 1.
results <- results - mean(results)
results <- results / sd(results)

# Plot the data.
hist(results)
```

We can employ the central limit theorem, and **sum up random (uniform) variables**.

There are much more efficient way, but we don't care here.

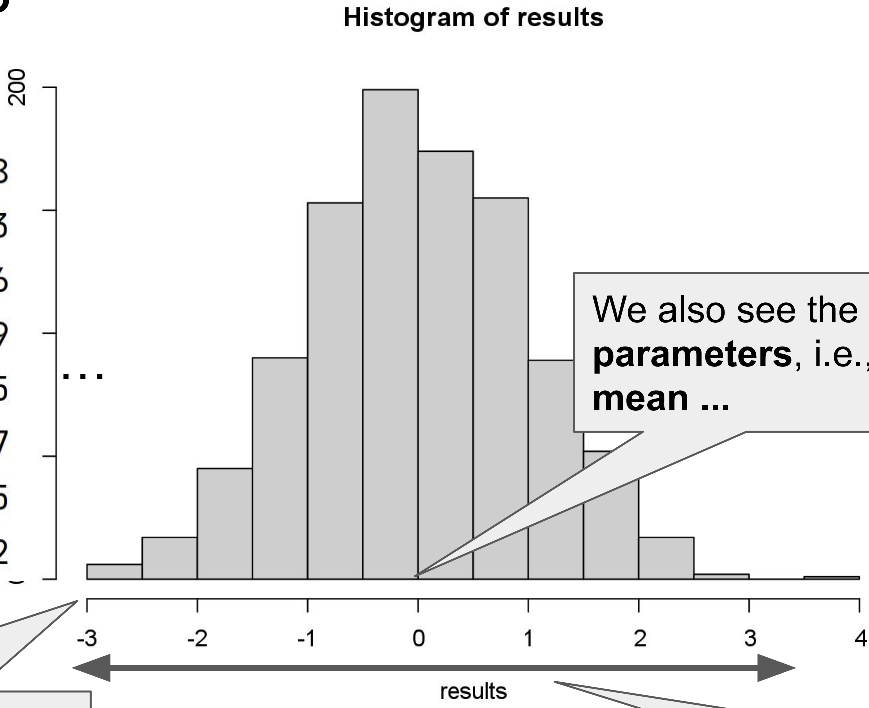
Plotting the results as histogram

```
 [1]  0.450729024 -0.809649934  0.764605888
 [7]  1.736888830  0.059764177 -0.576108443
[13]  0.419742595 -0.439459404 -1.177983766
[19] -2.586490148 -0.127243277  0.475076599
[25]  1.709972580  0.202269372 -0.471295545
[31]  0.612078333  0.998179839 -1.227958907
[37] -0.096025296  1.528426825 -0.065776675
[43]  0.243227469 -0.489929905  0.012433712
```

...

⋮

The random values



We also see the two **parameters**, i.e., **mean ...**

We have sampled from a **normal distribution**.

… and **standard deviation**.

R’s API

Normal {stats} R Documentation

The Normal Distribution

Description

[redacted] random generation for the normal distribution with mean equal to `mean` and standard deviation equal to `sd`.

Here it is called *random generation for…*
(╯°□°)╯︵ ┻━┻

Usage

[redacted]

```
rnorm(n, mean = 0, sd = 1)
```

Random number generation for other types of distributions

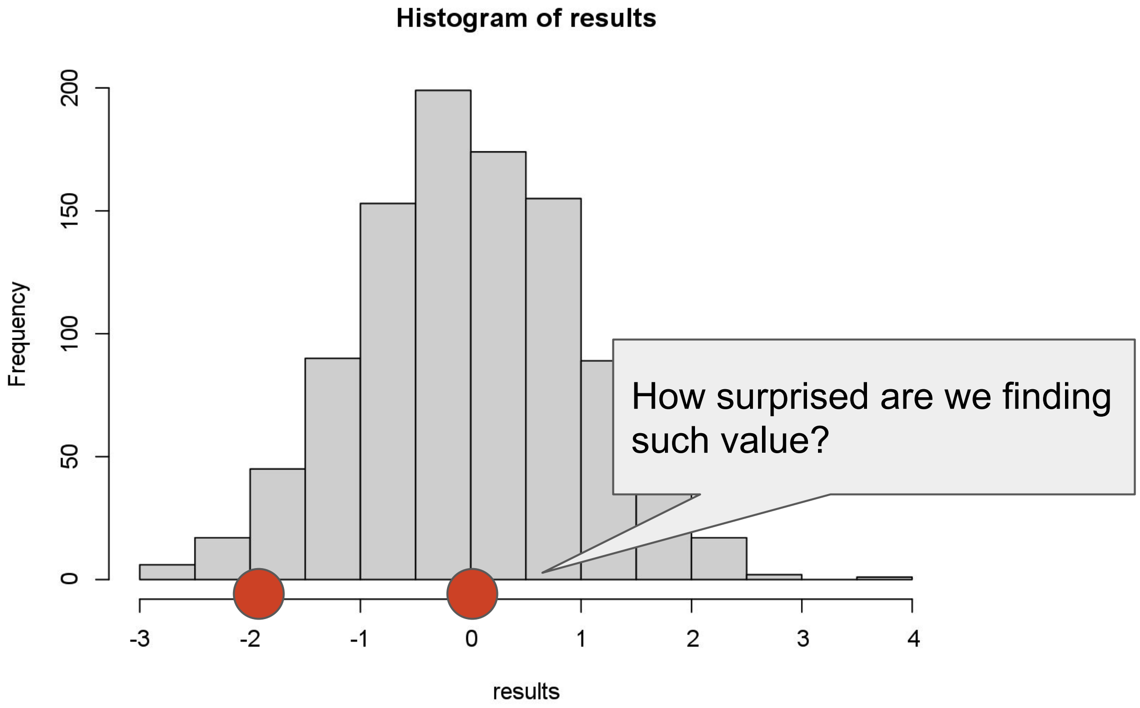
- Binomial distribution: `rbinom(n = ..., size = ..., prob = ...)`
- Poisson distribution: `rpois(n = ..., lambda = ...)`
- Exponential distribution: `rexp(n = ..., rate = ...)`
- …

Short Demo

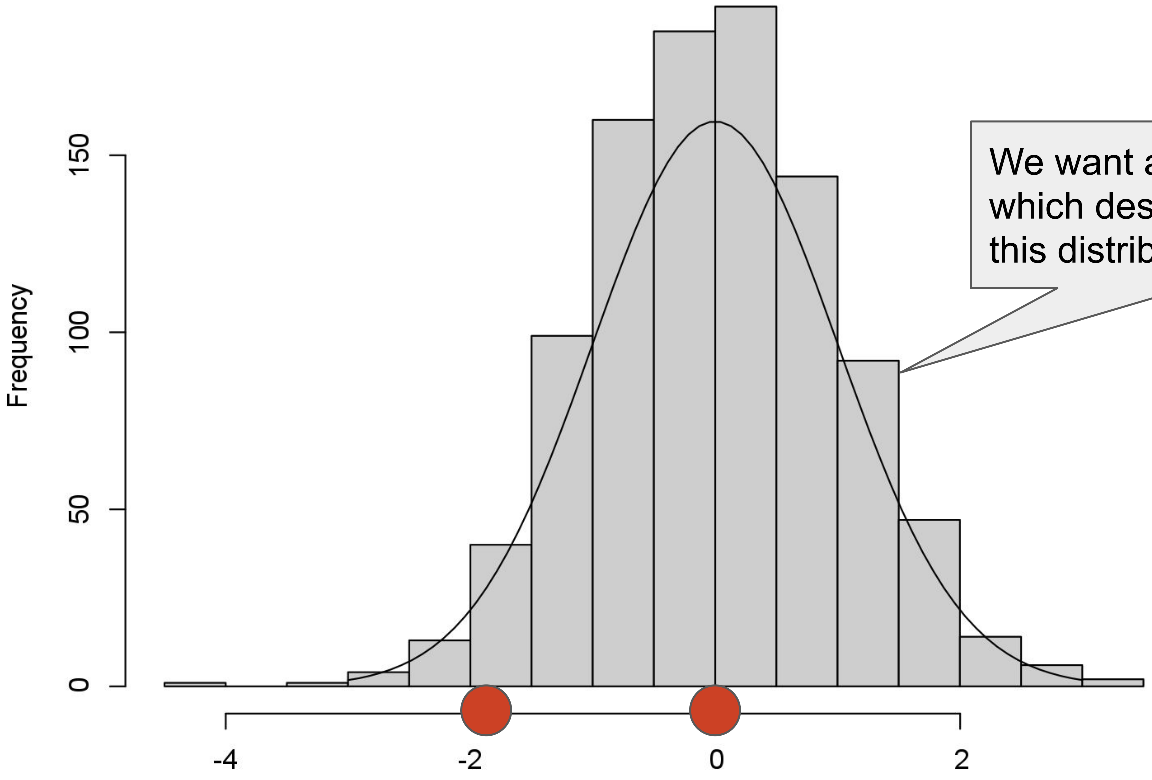
(The best way for understanding distributions is using them)

The **(probability) density** function (PDF)

We need the degree of being “surprised” by a value

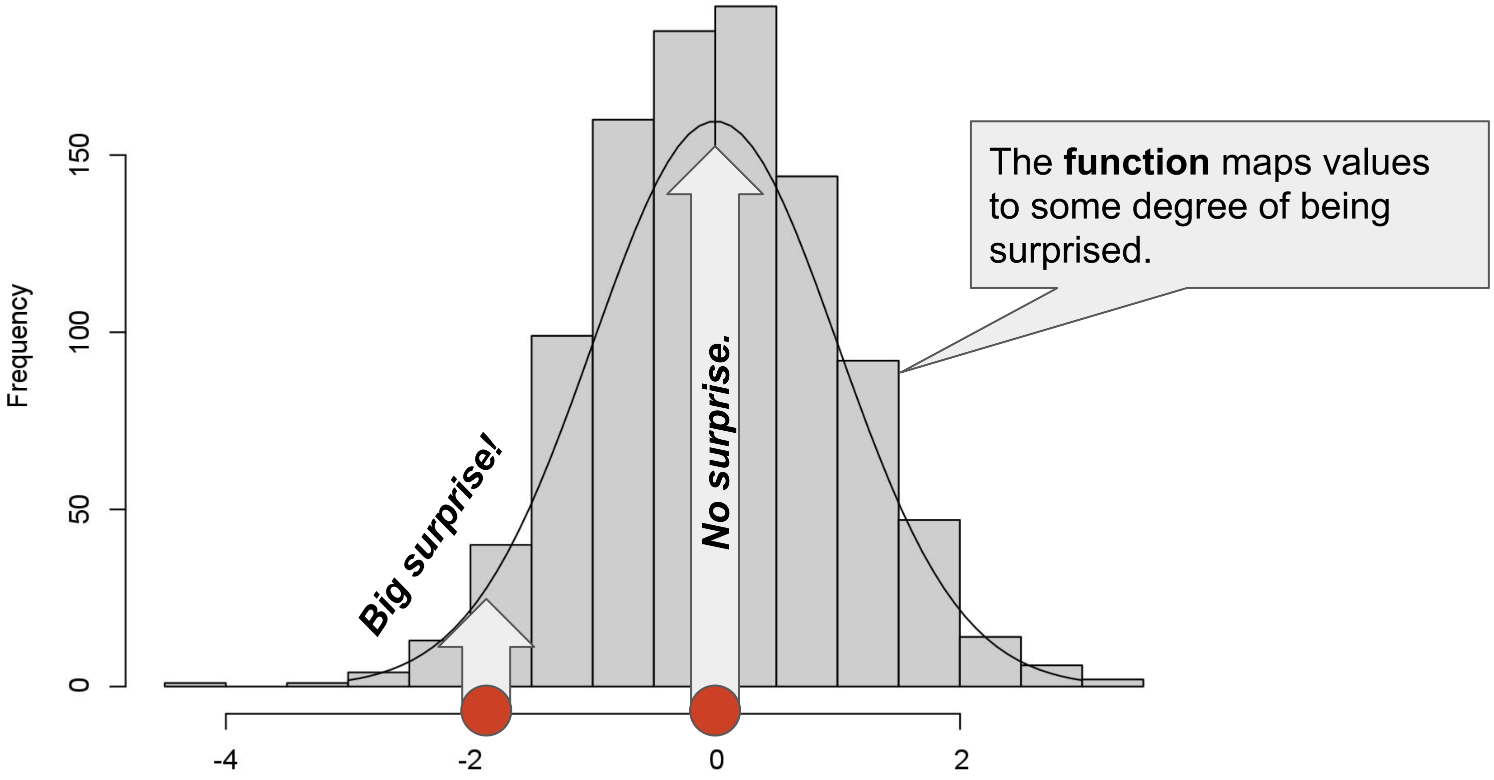


We need the degree of being “surprised” by a value (cont)



We want a function like that which describes the shape of this distribution.

We need the degree of being "surprised" by a value (cont)



The **(probability) density function (short PDF)**

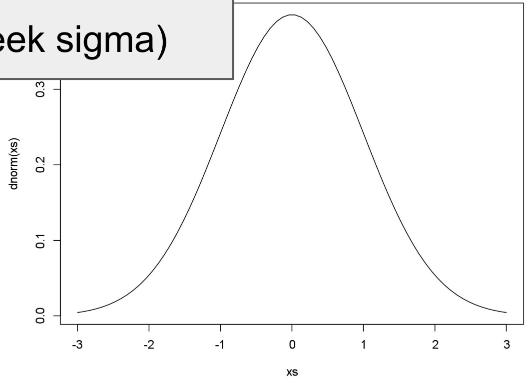
In this case, for the normal distribution (see, we have parameters again)

μ (greek mu) is the mean

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

σ (greek sigma)

Parameters are fixed again: σ (greek sigma) refers to the standard deviation.



R's API

We will never implement this by hand.

Normal {stats} R Documentation

The Normal Distribution

Just called density, but it's the probability density function (PDF).

Description

Density, [redacted] and random generation for the normal distribution with mean equal to `mean` and standard deviation equal to `sd`.

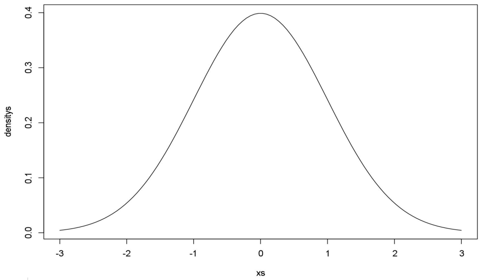
Usage

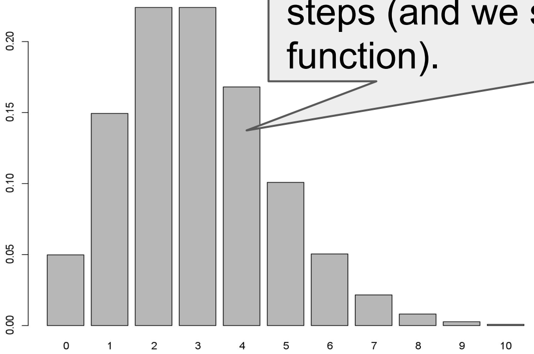
```
dnorm(x, mean = 0, sd = 1, log = FALSE)
[redacted]
rnorm(n, mean = 0, sd = 1)
```

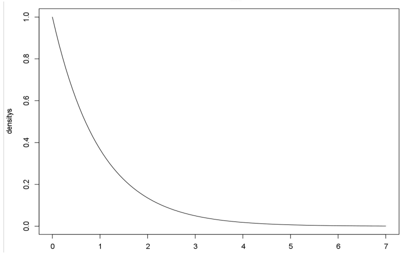
Probability density functions (PDF) for other types

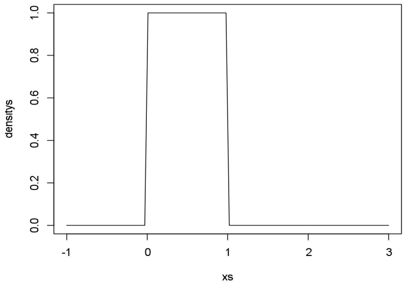
You might start to notice the symmetry (and where to find this functions).

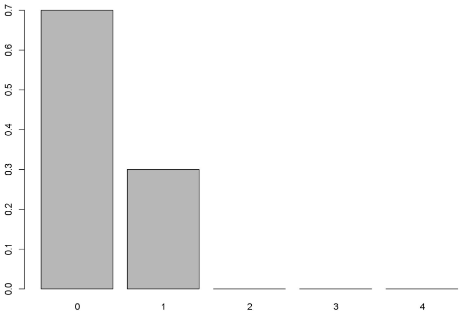
For discrete distributions, we have steps (and we say probability mass function).







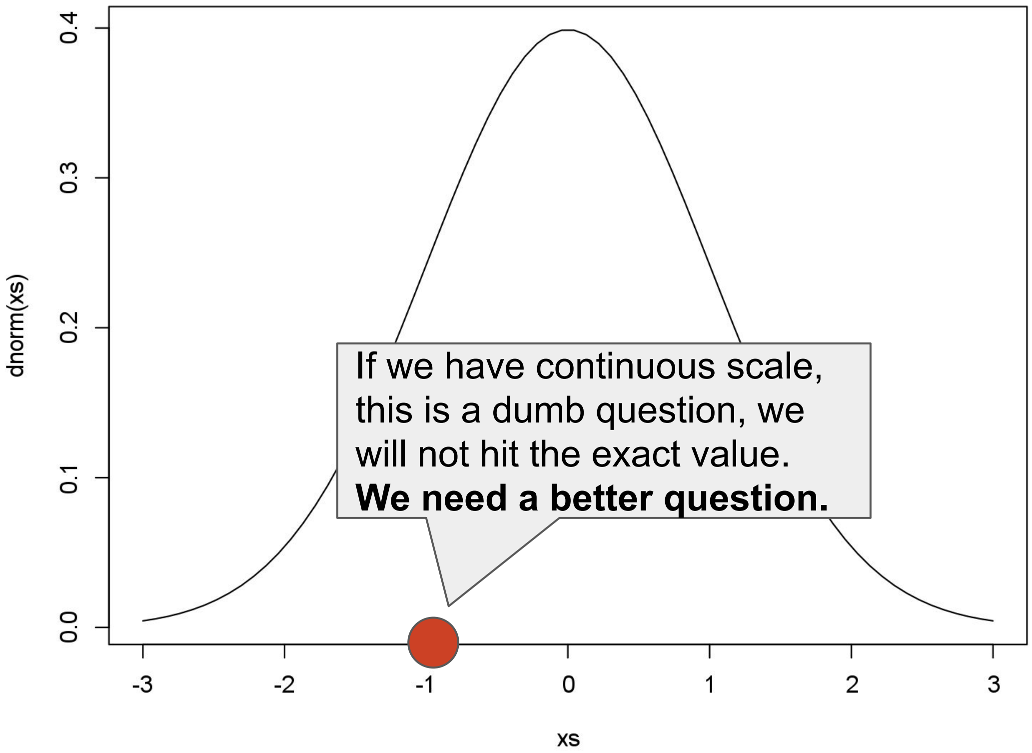




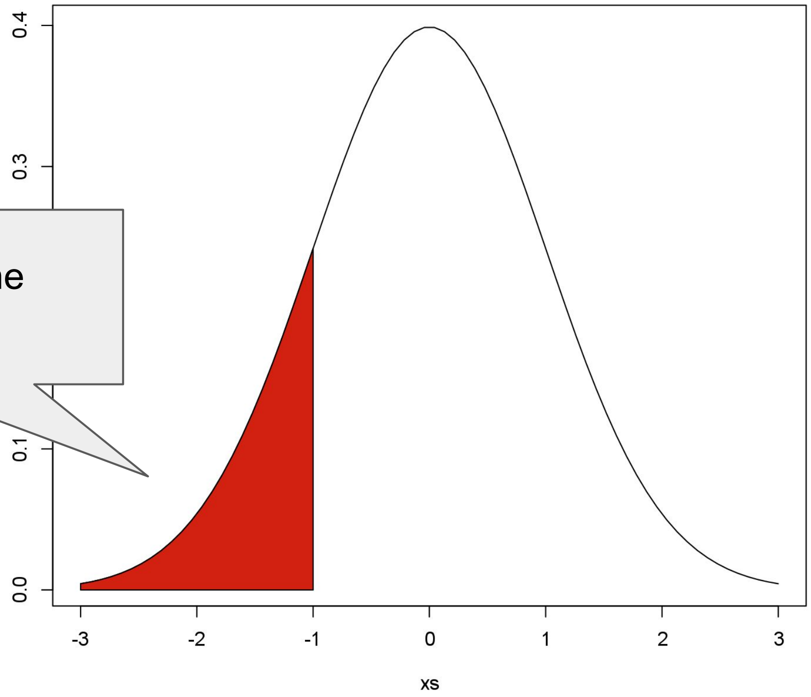
Short Demo

The **cumulative distribution** function (CDF)

What is the probability of facing an exact value?



What is the probability of facing a value within a certain range (e.g., X ≤ -1)?



Our question is about the **area under the curve**.

The cumulative distribution function (CDF)

for a normal distribution

$$F_X(x) = \underbrace{\int_{-\infty}^{x}}_{\text{new}} \underbrace{\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}}_{\text{PDF (in this example of normal distribution)}}$$

No **closed form solution for this case**; we need to approximate it numerically.

μ (greek mu) is the mean

σ (greek sigma)

If you have a discrete distribution, the integral can be replaced by a sum.

Parameters are fixed again: (greek sigma) refers to the standard deviation.

Since we may have no closed form solution, we approximate it.

Seen an example for the CDF of a normal distribution

```
# Checking the cumulative distribution function (CDF) for x smaller
or equal to 0.
x <- 0

values <- rnorm(n = 2000, sd = 1, mean = 0)

print(sum(values < x) / 2000)
```

Just generate 2000 random values and then do simple counting.

≈ 0.5

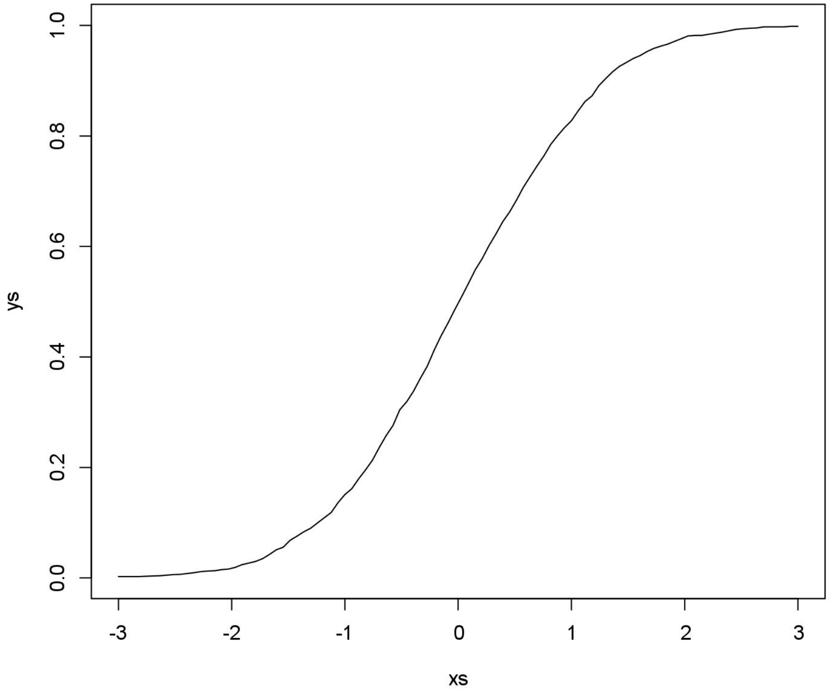
R can sum up boolean values (True = 1, False = 0)

Exploring the CDF with respect to X

```
# Plotting the cumulative distribution function (CDF) for x smaller or equal.
values <- rnorm(n = 2000, sd = 1, mean = 0)

ys <- NULL
xs <- seq(-3, 3, length.out = 100)
for (x in xs) {
 ys <- c(ys, sum(values < x) / 2000)
}

plot(xs, ys, type = "l")
```

The classical plot of the cumulative distribution function (CDF)

for the normal distribution



R’s API

Normal {stats} R Documentation

The Normal Distribution

Description

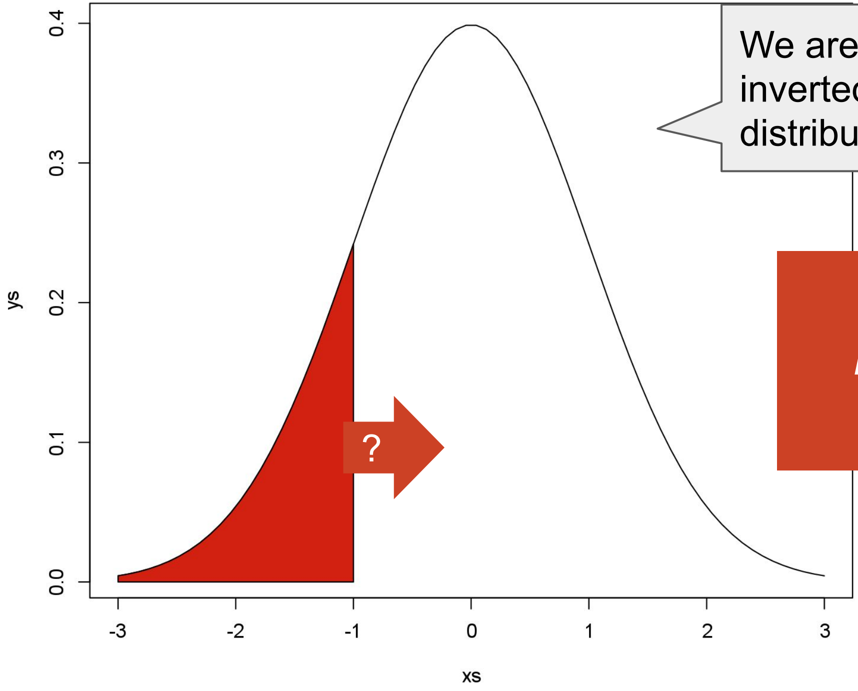
Density, distribution function [redacted] and random generation for the normal distribution with mean equal to `mean` and standard deviation equal to `sd`.

Usage

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

The **quantile** function

Requirement: Given the area, find the boundary:

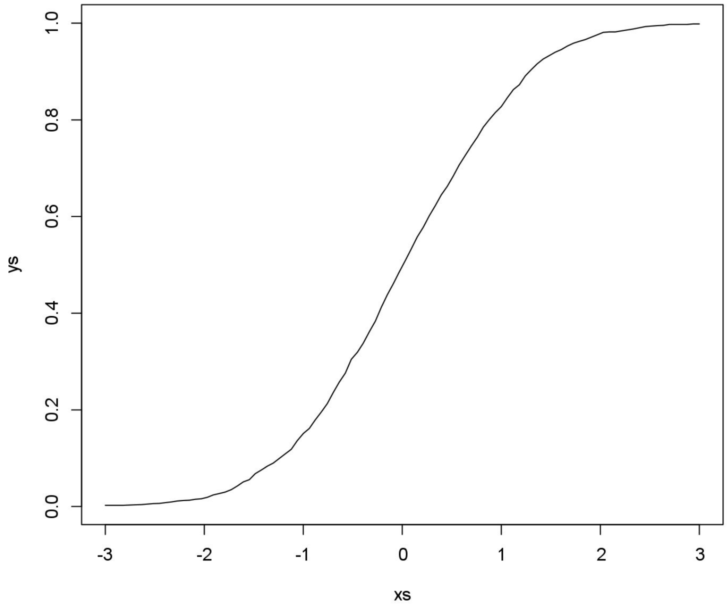


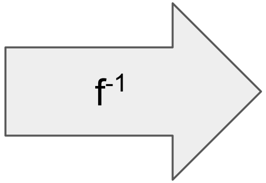
We are just searching for the inverted cumulative distribution function (CDF)

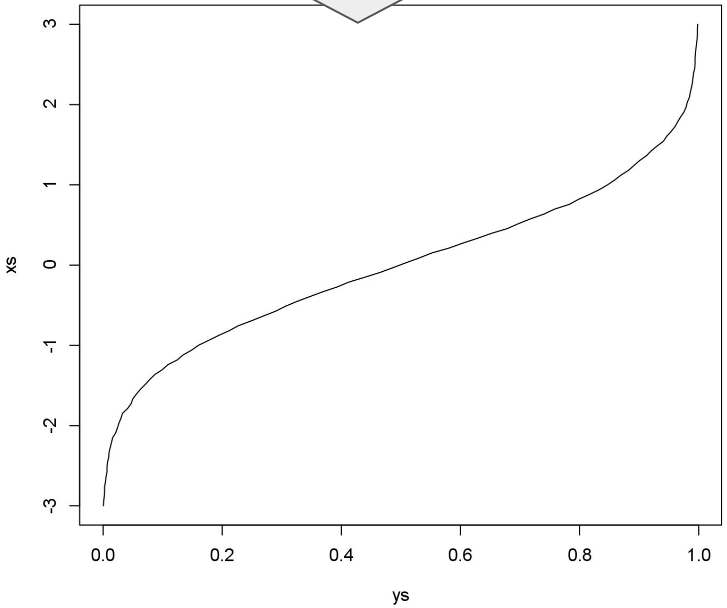
FIXED

Just inverting the CDF

BTW: x and y are just switched in the plot. `plot(ys,xs,...)`







R’s API

Normal {stats} R Documentation

The Normal Distribution

Description

Density, distribution function, quantile function and random generation for the normal distribution with mean equal to `mean` and standard deviation equal to `sd`.

Usage

```
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)
```

Example discrete distribution (binomial)

Binomial {stats} R Documentation

The Binomial Distribution

Description

Density, distribution function, quantile function and random generation for the binomial distribution with parameters `size` and `prob`.

This is conventionally interpreted as the number of 'successes' in `size` trials.

Usage

```
dbinom(x, size, prob, log = FALSE)
pbinom(q, size, prob, lower.tail = TRUE, log.p = FALSE)
qbinom(p, size, prob, lower.tail = TRUE, log.p = FALSE)
rbinom(n, size, prob)
```

Summary

- We have covered distributions in terms of **types, parameters and some interesting function**.
- What we are missing is a way to **come from fixed data to unknown parameters.** We will cover this next week.

We will meet all these distributions again in the remainder of this course.