SIT718 Real World Analytics

Lecturer: Dr Ye Zhu

School of Information Technology
Deakin University

Transforming data - Week 4

Transforming data

Learning aims:

- Understand the different roles that transformation of variables can play in pre- and post-processing of data;
- Introduce the Power means, which generalise the geometric, harmonic and arithmetic mean
- Build intuition about using transformations appropriately
- Using R to transform data

Read Chapter 2 of the reference book (An Introduction to Data Analysis using Aggregation Functions in R by Simon James)

Revision: Chapter 4. Distributions Stats Data and Models. De Veaux, Velleman, Bock, 4th edition, Pearson 2016.

WHY DO WE NEED TO TRANSFORM DATA?

Scale

Student	Sprint	Height	Serving	Endurance	AM
Mizuho	15.78	148	94	17	68.55
Yukie	21.15	147	94	20	70.43
Megumi	14.30	134	91	17	64.36
Sakura	19.59	174	88	16	74.50
Izumi	10.96	145	93	16	66.37
Yukiko	19.17	158	83	12	68.06
Yumiko	18.35	157	99	20	73.44
Kayoko	14.09	177	82	23	73.92

Megumi has the lowest score here - but that's largely due to her height. Not only are the heights higher, but importantly they are much more variable.

WHY DO WE NEED TO TRANSFORM DATA?

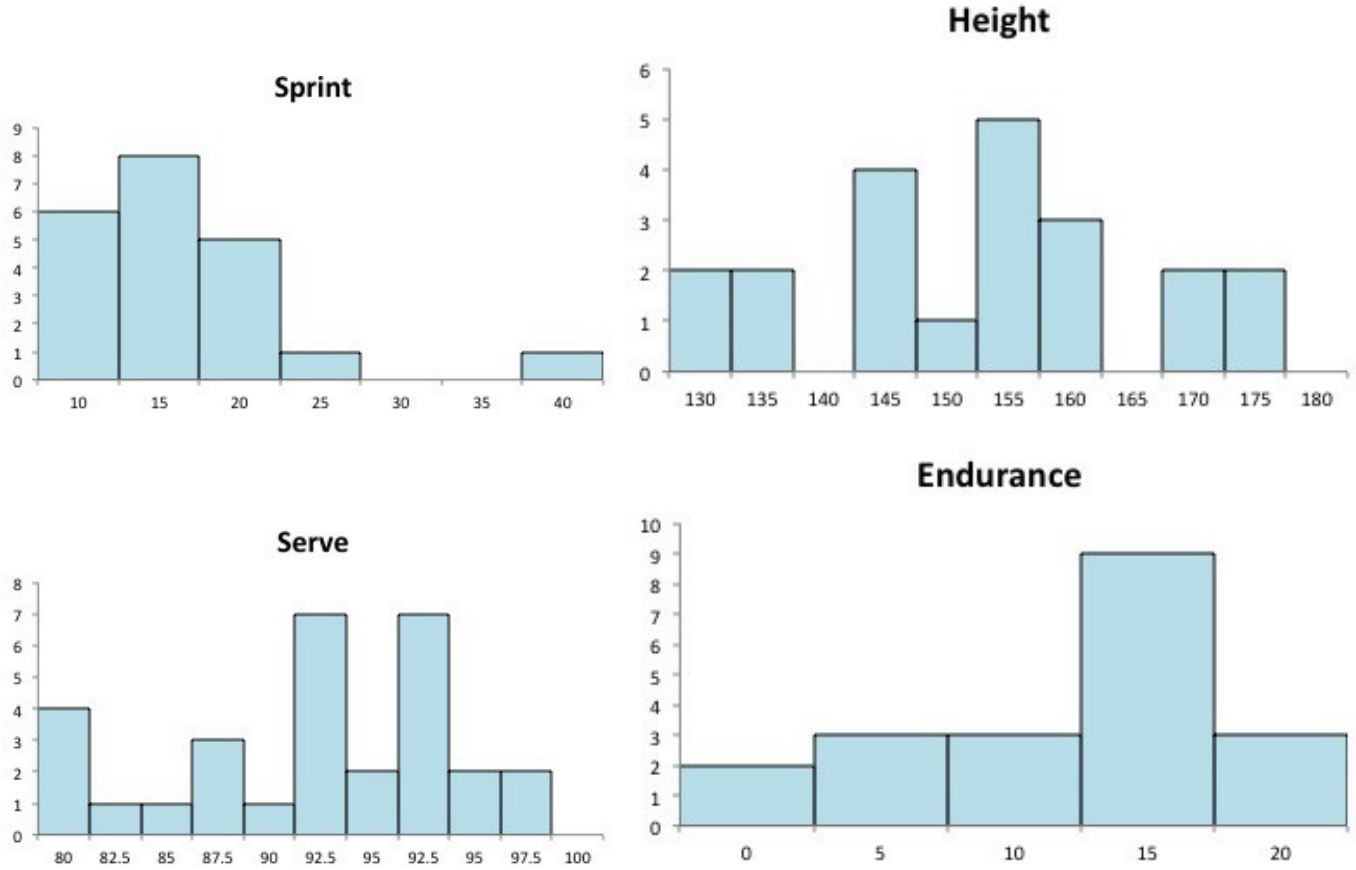
Raw score or utility?

Student	Sprint	Height	Serving	Endurance	AM
Mizuho	15.78	148	94	17	68.55
Yukie	21.15	147	94	20	70.43
Megumi	14.30	134	91	17	64.36
Sakura	19.59	174	88	16	74.50
Izumi	10.96	145	93	16	66.37
Yukiko	19.17	158	83	12	68.06
Yumiko	18.35	157	99	20	73.44
Kayoko	14.09	177	82	23	73.92

In this case, we want to reward *lower* sprinting times!

WHY DO WE NEED TO TRANSFORM DATA?

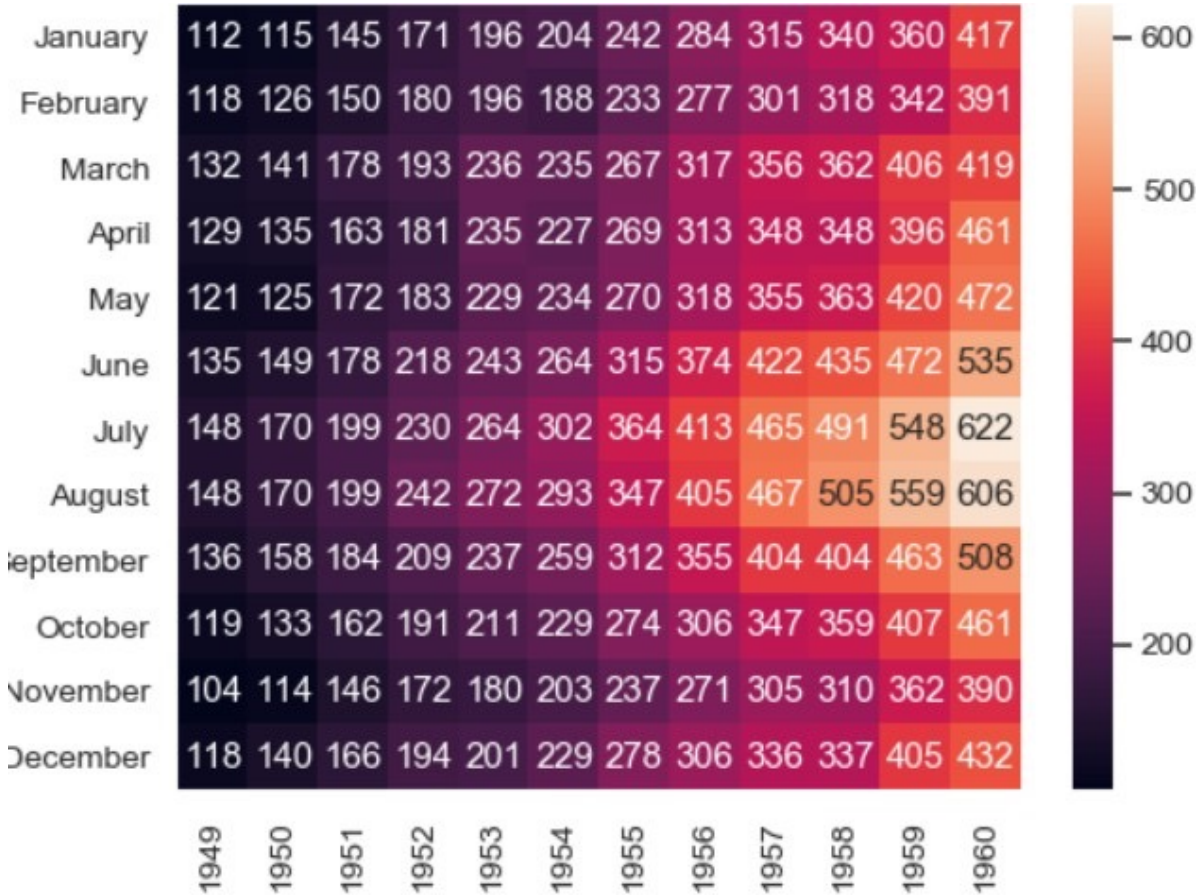
Distribution



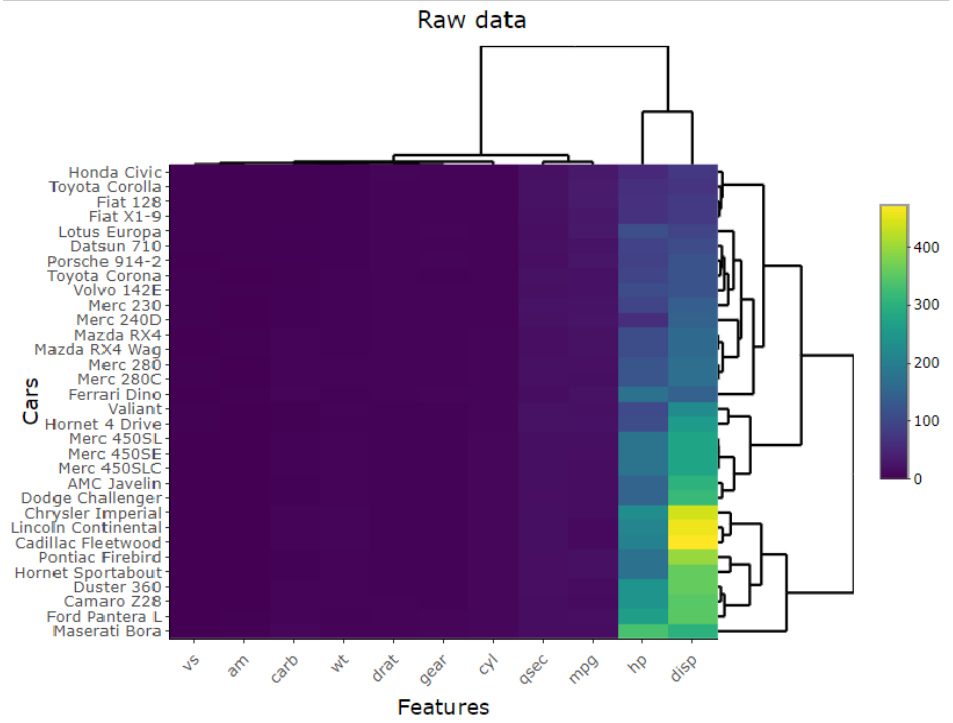
Correcting for distributions can be very difficult. There are some standard approaches we can use, however none of these is fool-proof and so we are often required to make a judgment call about what is reasonable.

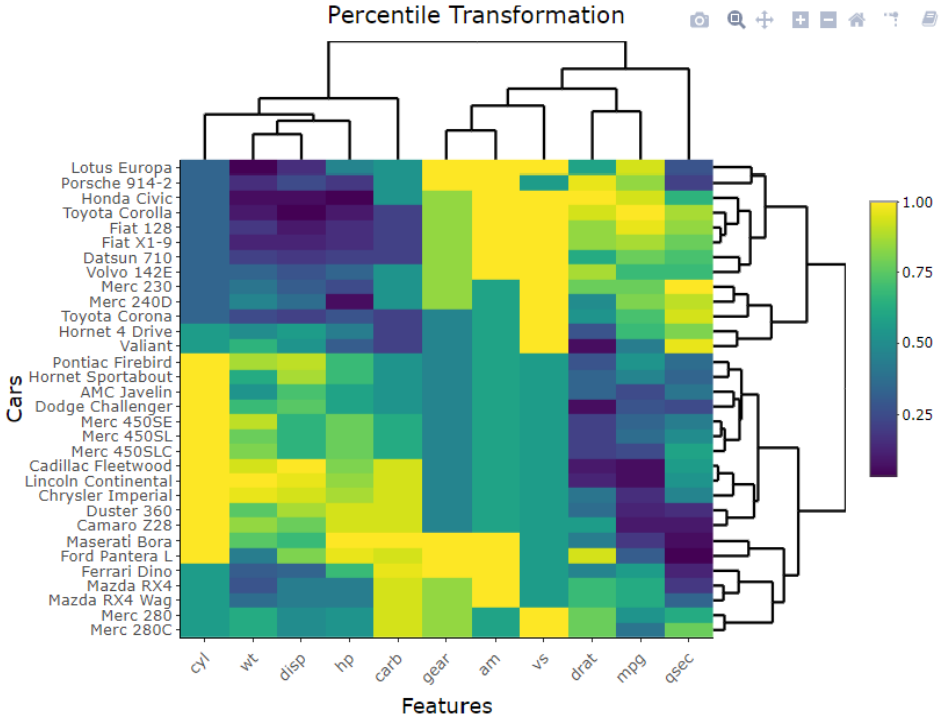
WHY DO WE NEED TO TRANSFORM DATA?
Heat Map Example

A heat map (or heatmap) is a graphical representation of data where values are depicted by color. Heat maps make it easy to visualize complex data and understand it at a glance



WHY DO WE NEED TO TRANSFORM DATA?
Heat Map Example (continue...)





https://www.datanovia.com/en/blog/how-to-normalize-and-standardize-data-in-r-for-great-heatmap-visualization/

Why do we need to transform data?

So essentially we should be considering the following:

- If we are using *aggregation functions*, we need to bear in mind the interpretation of monotonicity. We need to ensure that it makes sense that an increase in the input should result in an increase to the output.
- **Scale** and distribution of the data may differ. We may want/need to have all the data transformed to a particular interval, and we also might want to to ensure that increases are treated similarly no matter where they are on the interval.
- Specific to averaging aggregation functions, we might want to consider the role that idempotency of the function plays.
- The data might not be numeric at all (e.g., ordinal variables). If we assign numeric values, are these reasonable and justified?

NEGATIONS AND UTILITY TRANSFORMATIONS

- Negation functions transform the data so that *high values become low and low values become high*.
- For the unit interval (between 0 and 1), the **standard negation** is given by

$$N(t) = 1 - t.$$

- When defined over a particular interval, a *strong negation* is one which satisfies the property of *involution*, i.e. if we perform a negation of the negation then we get the original value.

$$N(N(t)) = t.$$

Negations and utility transformations

We need to pay attention to the values over which our data is distributed.

For the girls' volleyball team, we wanted to transform the Sprint variable. The data ranges from 10 to 40. So we can use a negation:

$$N(t) = 40 - t + 10.$$

In general, if our interval is $[a, b]$, we can use

$$N(t) = b - t + a$$

$$N(t) = a + b - t$$

Does this satisfy involution?

Let's check:

$$N(N(t)) = b - N(t) + a$$

$$= b - (b - t + a) + a = b - b + t - a + a = t$$

Negations and utility transformations

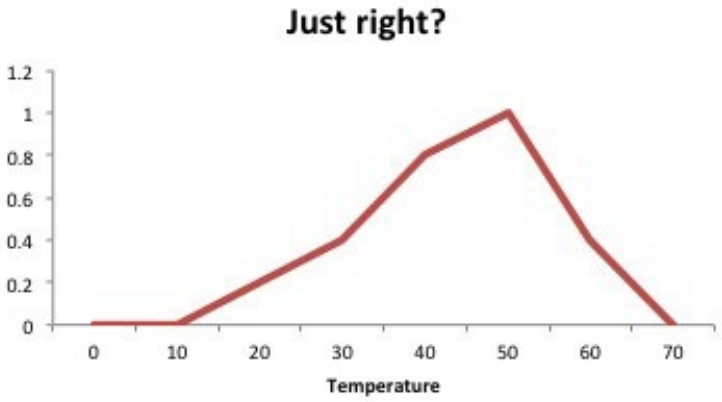
We can have other negations too.

- Formally, for the interval [0, 1]a negation is any function that satisfies $N(1) = 0$, $N(0) = 1$ and is monotone decreasing.
- If we have a general interval, this would become $N(a) = b$, $N(b) = a$.
- Remember that if it's a **strong** negation, then we require $N(N(t)) = t$.

Negations and utility transformations

Utility/suitability transformation:

- Rather than just our data going in the wrong direction, we might have situations where 'good' might actually refer to some **intermediate value**, and values get worse as they go further away.
- E.g. In judging the suitability of porridge temperature, Goldilockes might consider the following.



Negations and utility transformations

- Whether we use the standard negation or something more complicated, there aren't necessarily set rules here.
- The standard negation will ensure that most of the features of the distribution are preserved, we are essentially just flipping the data so that it is consistent with the other variables.

SCALING, STANDARDISATION AND NORMALISATION

The simplest and most common technique we might employ to ensure that variables taking values over different ranges can be aggregated is to apply scaling transformations.
To get all values to the interval [0, 1]we can use:

$$x_{\text{new}} = \frac{x_{\text{old}} - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$$

This is sometimes referred to as feature scaling.

Scaling, standardisation and normalisation

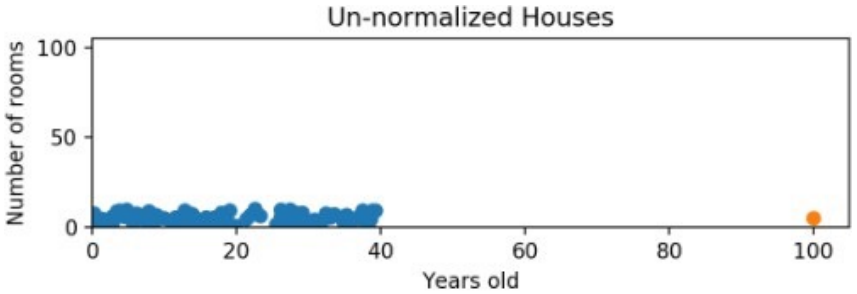
In statistics you will also have come across the idea of standardisation or Z-scores.

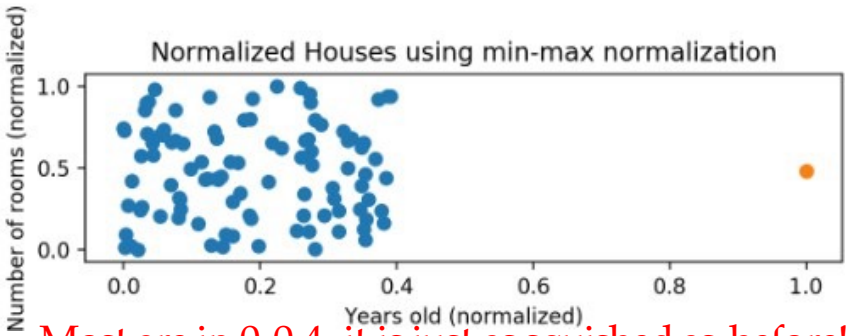
$$x_{\text{new}} = \frac{x_{\text{old}} - AM(\mathbf{x})}{\sigma}$$

where σ is the standard deviation.

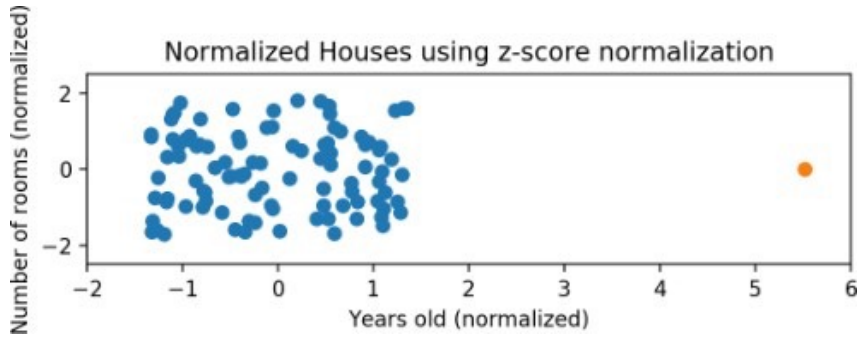
- This is an appropriate transformation if our data all follow a normal distribution but with different means and standard deviations.
- We could then similarly transform to the unit interval as we did previously.

SCALING, STANDARDISATION AND NORMALISATION





Most are in 0-0.4, it is just as squished as before!



Min-max normalization:
Guarantees all features will have the exact same scale but does not handle outliers well.

Z-score normalization:
Handles outliers, but does not produce normalized data with the exact same scale.

https://www.codecademy.com/articles/normalization#:~:text=Min%2Dmax%20normalization%3A%20Guarantees%20all,with%20the%20exact%20same%20scale.

SCALING, STANDARDISATION AND NORMALISATION

Assuming there are no extreme outliers, one transformation that in most cases will allow normally distributed data to be scaled to the unit interval is:

$$x_{\text{new}} = 0.15 \left(\frac{x_{\text{old}} - AM(\mathbf{x})}{\sigma} \right) + 0.5$$

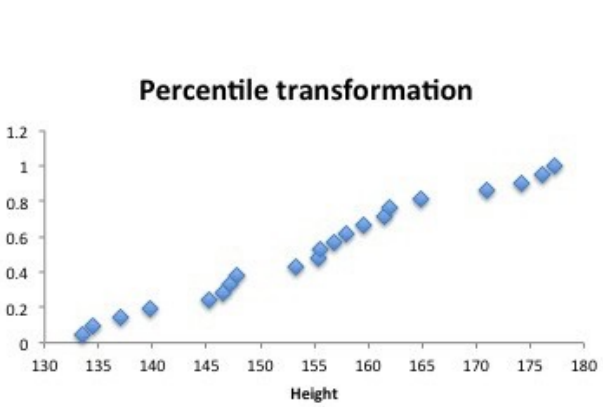
The 0.15 multiplying the usual standardization formula will first scale the data so that 99.7% of the data should fall between -0:45 and 0:45 (three standard deviations) and then adding 0.5 will shift the interval to [0.05,0.95]

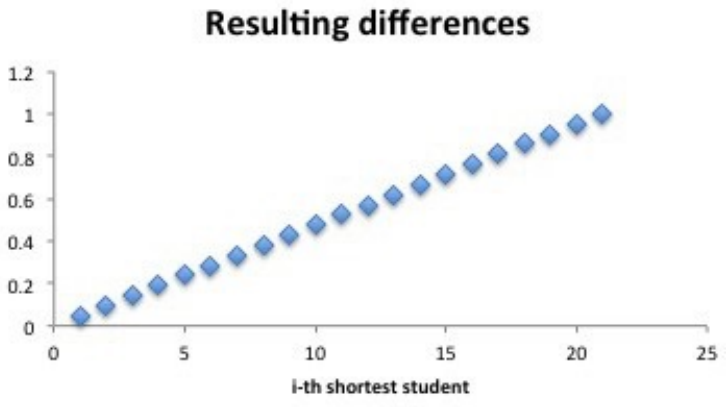
Scaling, standardisation and normalisation

Both of these transformations can be considered as a kind of normalisation. However there are other options too.

Rank-scaling:

- We might want the data to properly reflect percentiles or quartiles.
- The final score in year 12 used as a basis for applying to universities uses this approach. Essentially, the students' combined study scores are used to **put them in order,** and then a percentile score is allocated so that a score of 74.35 means that a student scored better than 74.35% of all students.





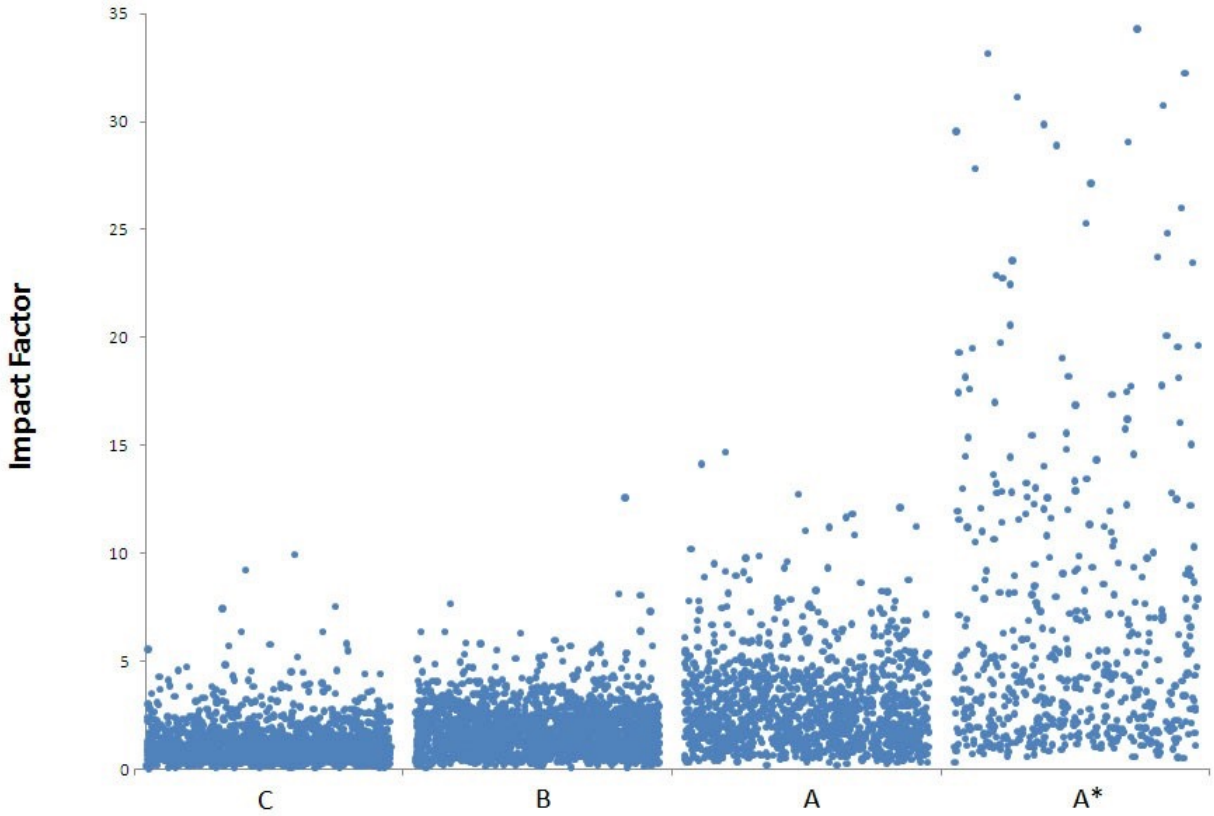
Log and polynomial transformations

- Another common technique in statistical regression is to transform the data using functions like $x_{\text{new}} = \ln(x_{\text{old}})$ and $x_{\text{new}} = (x_{\text{old}})^2$.
- Such functions can help if data are exponentially distributed or skewed.
- The log function in particular is useful for data that can have a few very large inputs.
- Incomes, populations in ecology and academic journal citations are all examples of data that could exhibit such properties.

LOG AND POLYNOMIAL TRANSFORMATIONS

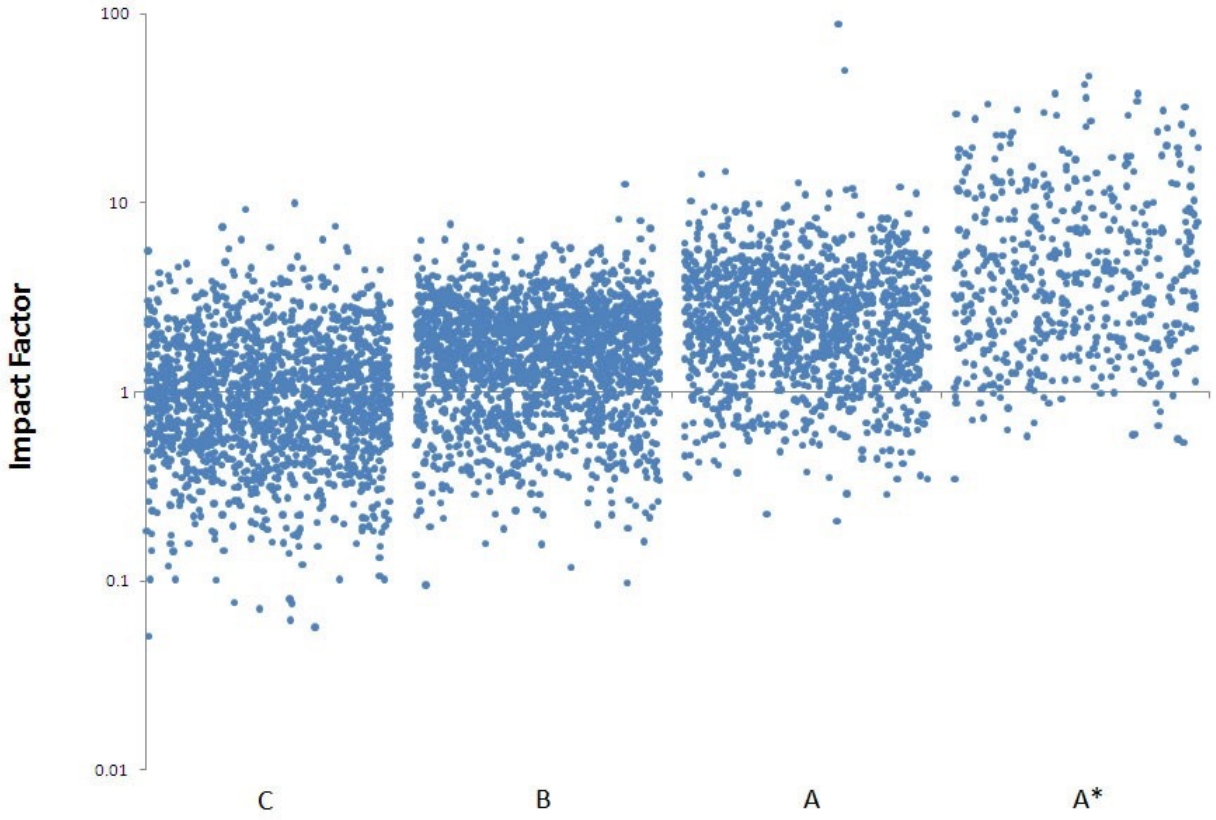
Showing the distribution of journal 'impact factors' according to Australian research council ranking.

impact factors: the ratio of citations to a journal's papers within the last two years to articles published



Log and polynomial transformations

- Showing the distribution after applying a log transformation.
- Although the general difference between the values is somewhat maintained, values toward the lower end of the scale become more spread out while higher values are pushed closer together.



LOG AND POLYNOMIAL TRANSFORMATIONS

- Polynomial functions (such as t^2 or $t^{\frac{1}{2}}$) can have a similar (but less dramatic) affect, allowing *skewed distributions to look more like normal distributions*.
- For t^p, $0 < p < 1$ can be used when there are fewer very high values (**positive skew**), while $p > 1$ can be used if the majority of data is gathered in the high range with fewer very low values (**negative skew**). This transforms work well when the data is already on the unit interval.

A note about an Error in the reference book:
Note that in the reference book, the above statement is mentioned in the wrong way around (page 52, first paragraph). The above statement is the correct one.

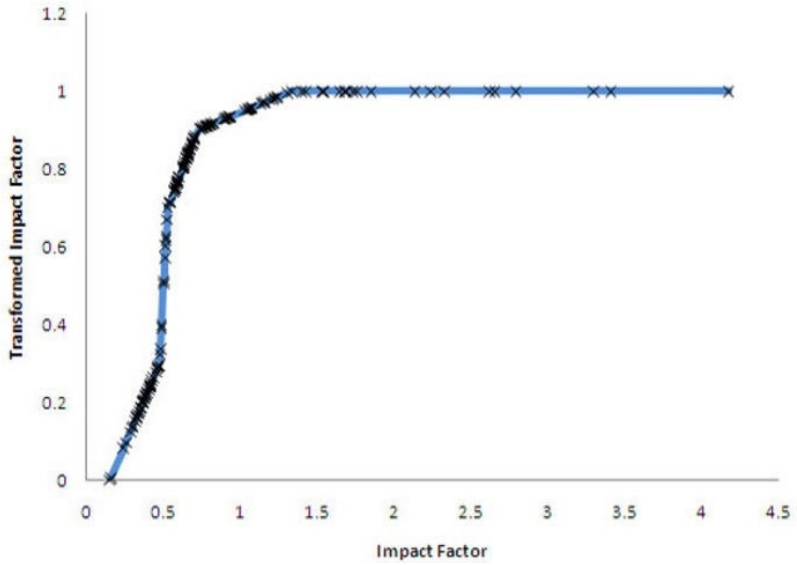
Piecewise-Linear Transformations

Piecewise-Linear Transformations

- construct a transformation from linear 'pieces'.
- The following was used with the journals data set so that the quartiles corresponded with certain values along the interval, e.g. if the journal was in the top 5% then it should have a score of between 95 and 100.
- In this ranking exercise, the A* ranked journals were considered to be in the top 5% of journals, the A ranked journals were in the top 20%, B in the top 50% and C otherwise
- The domain was split into sub-intervals corresponding with the median impact factor score for each journal and the values in the unit interval these values were chosen so that the percentiles values would fall between them, i.e. $y_C = 0.3$, $y_B = 0.7$, $y_A = 0.9$.

PIECEWISE LINEAR TRANSFORMATION

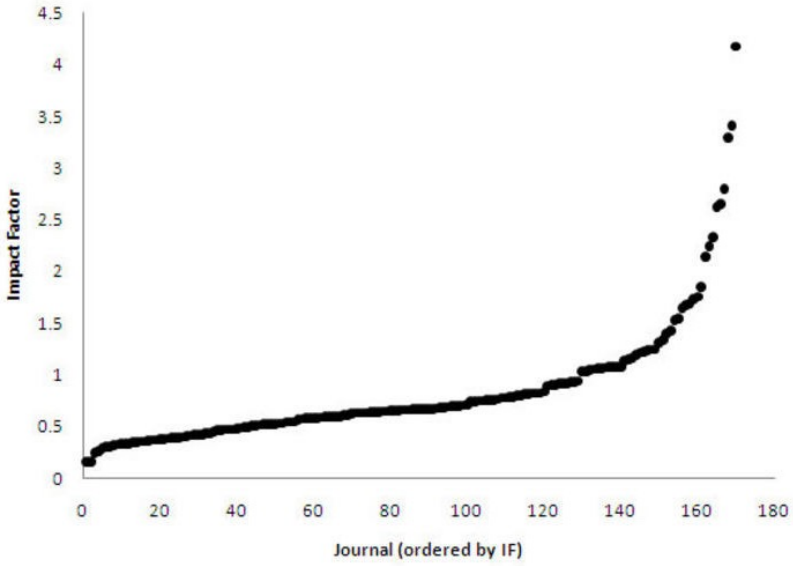
A piecewise linear transformation.

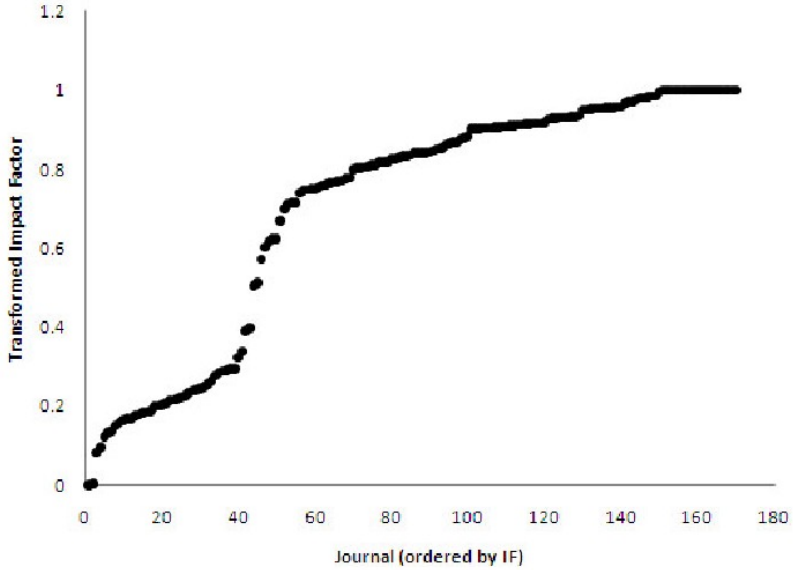


$$x'_j = \begin{cases} y_C \frac{x_j - \min(\mathbf{x}_j)}{C_{Med} - \min(\mathbf{x}_j)}, & x_j < C_{Med}; \\ y_C + (y_B - y_C) \frac{x_j - C_{Med}}{B_{Med} - C_{Med}}, & x_j < B_{Med}; \\ y_B + (y_A - y_B) \frac{x_j - B_{Med}}{A_{Med} - B_{Med}}, & x_j < A_{Med}; \\ y_A + (y_{A*} - y_A) \frac{x_j - A_{Med}}{A*_{Med} - A_{Med}}, & x_j < A*_{Med}; \\ y_{A*} + (1 - y_{A*}) \frac{x_j - A*_{Med}}{1 - A*_{Med}}, & \text{otherwise.} \end{cases}$$

PIECEWISE LINEAR TRANSFORMATION

Distribution before and after using piecewise transformation.





Ordinal inputs

Be careful if converting ordinal (or non-numeric) data to numeric data!

- For example, it is common to take Likert scale type questions "strongly disagree, disagree, agree, strongly agree" etc, and give each of these a numeric value.
- It can be very dangerous to draw inferences from such data, because people don't usually think on a linear scale. The real difference between disagree and neutral could be very different to the difference between neutral and agree or agree and strongly agree. Furthermore, people often don't think on the same scale.
- If you're going to transform such data for purposes of aggregation or regression, make sure you treat the results with this kind of thing in mind.

Functions built from transformations

In fact we can use transformations to build new functions.

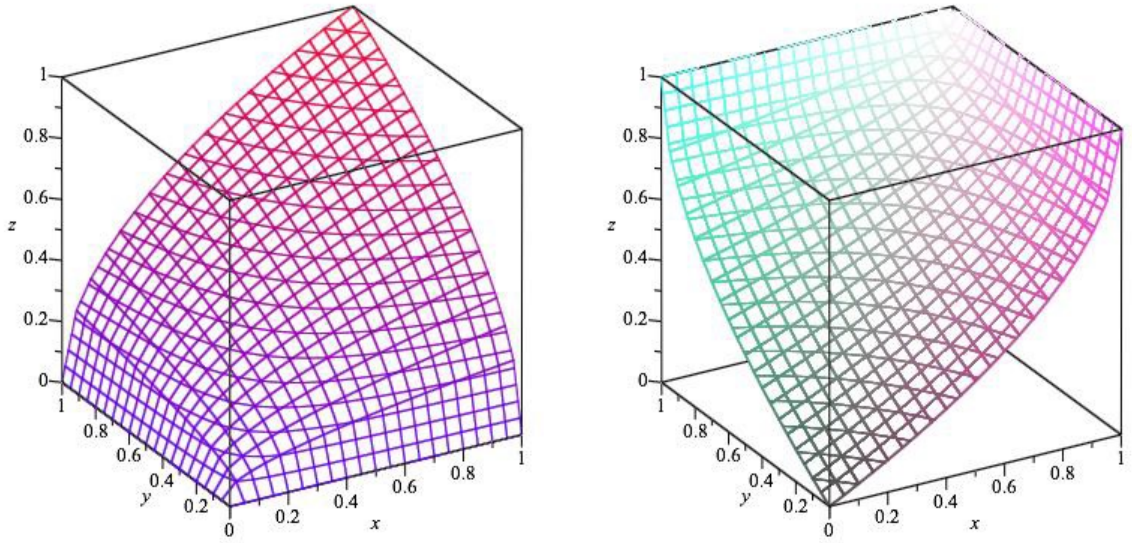
- One such type of function is called the **dual** aggregation function, which is built from a negation.
- For a given aggregation function A, it's dual A^d is given by:

$$A^d(\mathbf{x}) = 1 - A(1 - x_1, 1 - x_2, \ldots, 1 - x_n)$$

- Let's see what happens when we take the dual of some of our important functions.

FUNCTIONS BUILT FROM TRANSFORMATIONS

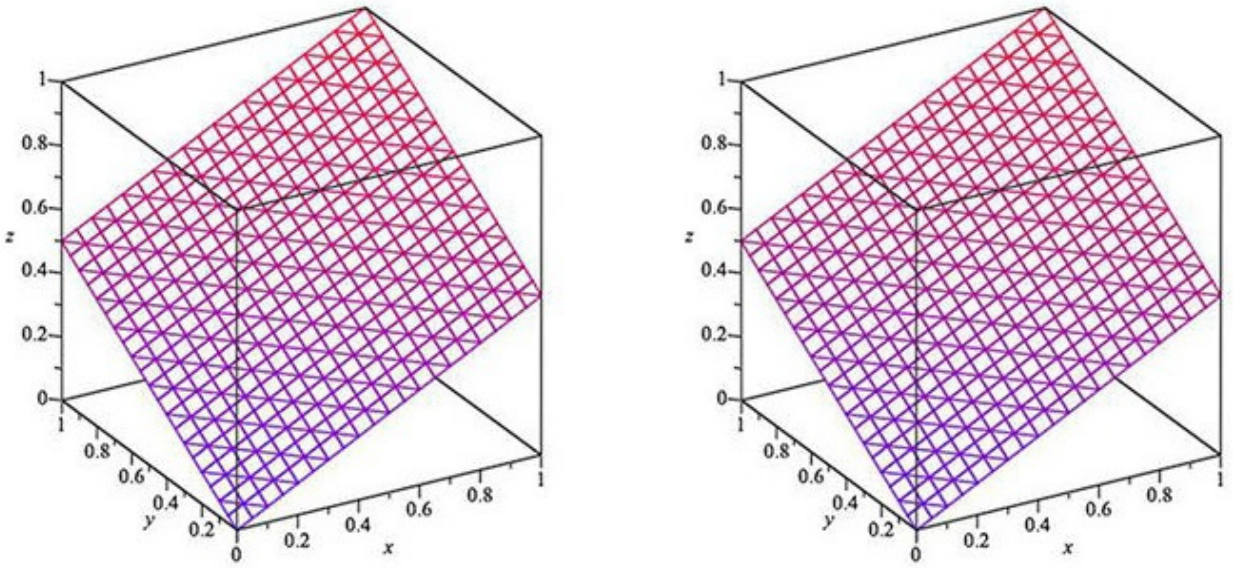
The geometric mean and it's dual GM^d.



- The dual function essentially exhibits reciprocal properties to the original function, but it remains monotone increasing.
- We note that the geometric mean of any set which includes $x_i = 0$ will have an output of zero.
- With its dual, any input set which includes $x_i = 1$ will automatically have an output of 1.

FUNCTIONS BUILT FROM TRANSFORMATIONS

The arithmetic mean and its dual AM^d.



Since the arithmetic mean exhibits uniform behaviour over the interval, its dual is actually same function.

Power means

- Before we saw that transforming data could be used to help address distributions that were skewed or included high values.
- There are special families of means built from these ideas.
- One such family is the power mean, it is based on a transformation $g(t) = t^p$.

Formula (The power mean)

$$PM_p(\mathbf{x}) = \left(\frac{1}{n} \sum_{i=1}^{n} x_i^p \right)^{\frac{1}{p}} = \left(\frac{x_1^p + x_2^p + \cdots + x_n^p}{n} \right)^{\frac{1}{p}}$$

Power means

Formula (The power mean)

$$PM_p(\mathbf{x}) = \left(\frac{1}{n} \sum_{i=1}^{n} x_i^p \right)^{\frac{1}{p}} = \left(\frac{x_1^p + x_2^p + \cdots + x_n^p}{n} \right)^{\frac{1}{p}}$$

-
  - For $p = 1$ we obtain the **arithmetic mean**
  - For $p = -1$ we have the **harmonic mean**
  - For $p = 0$ (a limiting case) we actually obtain the **geometric mean**.
  - as $p \to -\infty$ we approach the **minimum** function
  - as $p \to \infty$ we approach the **maximum**.
- The *power mean* is sometimes referred to as a *generalized mean*, since it includes many means as special cases.

Power means

Guided by our special cases, we can consider how the function will treat inputs depending on the value of p

- Larger values of p and > 1 will mean that the output of the function is more influenced by larger values in the input set than by smaller ones. So the output will be dragged towards higher inputs. When p becomes infinitely large, the output will simply be the highest input value.
- Values of p below 1 will drag the function towards lower inputs.
- Once we have $p \leq 0$, we always get functions that will go to zero if any of the inputs are equal to zero. e.g. we already saw that the geometric mean ($p = 0$) and harmonic mean ($p = -1$) had an absorbent element of 0). Values of $0 < p < 1$ will still tend toward lower values but won't have this absorbing element property

POWER MEANS

Example:
Compare the values of the power mean for $p = -5, 0, 1, 3$ for the input vector $x =< 0.3, 0.9, 1 >$.

Solution:
Calculating these values respectively gives outputs of approximately 0.373, 0.646, 0.733 and 0.837.
It is clear that while $p = -5$ pushes the output closer to the 0.3, as it is increased the output draws closer to the 1.

Quasi-arithmetic means

An even more general class are the quasi-arithmetic means. Rather than a transformation of the variable using $g(t) = t^p$, the function g can take any form, as long as it's an increasing and invertible single-variate function over the domain of the inputs.

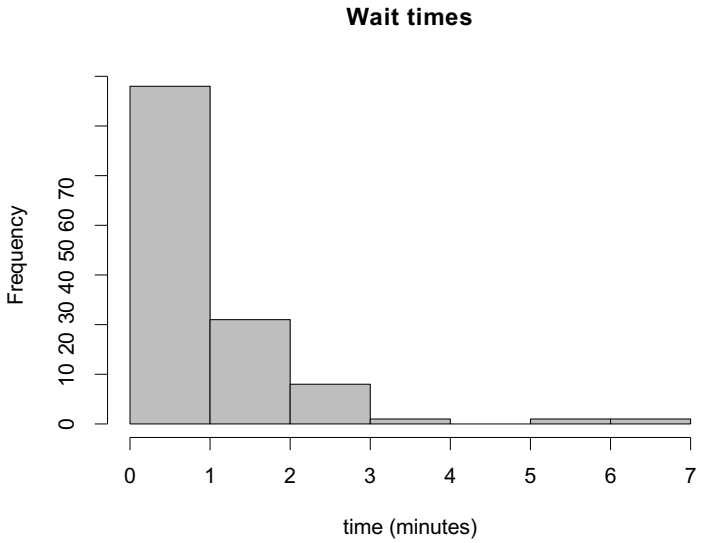
Formula (The quasi-arithmetic mean)

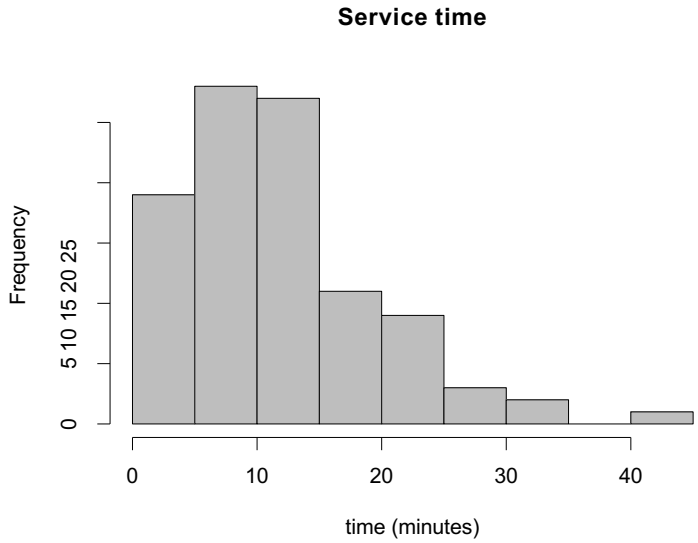
$$QAM_g(\mathbf{x}) = g^{-1}\left(\frac{1}{n}\sum_{i=1}^{n} g(x_i)\right)$$

Of course if g is t^p then we obtain the power means as a special case. The function g can even be a piecewise function like we saw before.

EXAMPLES

- Consider the following wait and service times for 100 customers.





What would be some potential transformations that could be used to scale both measurements to the unit interval?

EXAMPLES: Q1 SOLUTION

Depending on our purpose of aggregation. If we want to find the **average total time** then this actually should be done without any transformation of the inputs. On the other hand, if we are interested in evaluating the **customer experience based on wait time and service time**, then we probably need to apply scaling and negation (usually we'd say that a longer time is worse). We can also see from the distributions that both variables have more values closer to zero, so transformations like log, or x^p with p less than 1 could be useful. For example,

Wait times. Let the minimum be a and the maximum wait time be b. Denoting the original wait time by x and the transformed weight time by x':

$$x' = 1 - \frac{\ln x - \ln a}{\ln b - \ln a}$$

(this is equivalent to first transforming all the variables using log, then applying linear feature scaling, then applying the standard negation.)

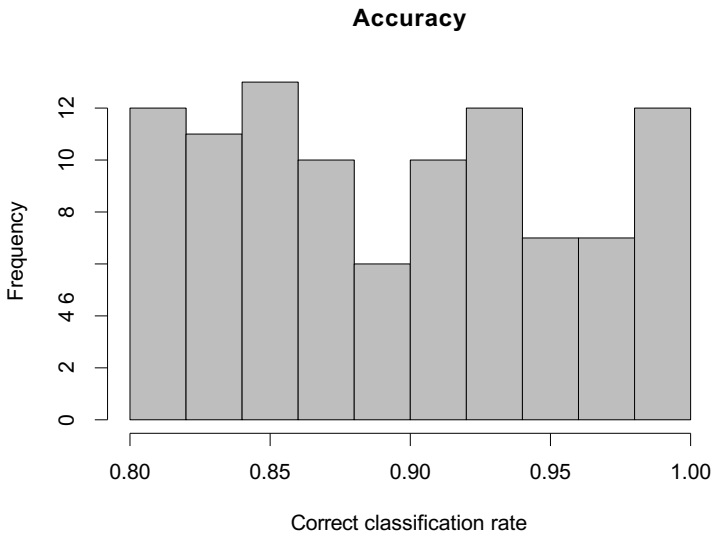
Service times. Let the minimum be a and the maximum wait time be b. Denoting the original wait time by x and the transformed wait time by x':

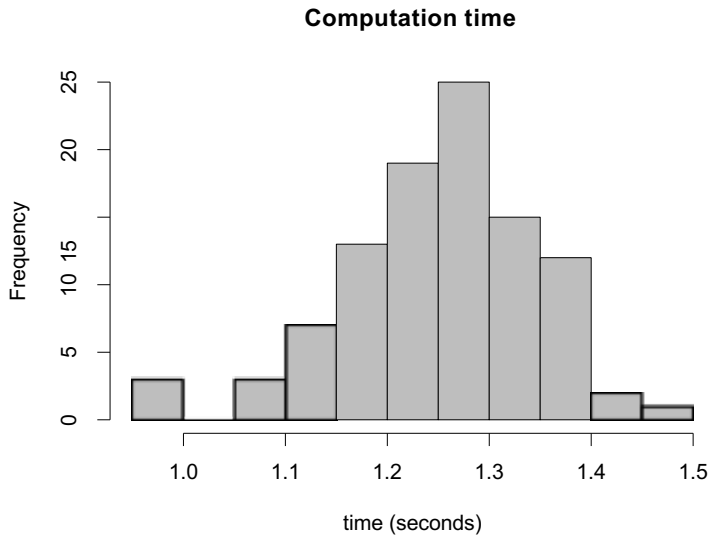
$$x' = 1 - \frac{x^{1/2} - a^{1/2}}{b^{1/2} - a^{1/2}}$$

(Similar to Wait times however using a square root instead of log since the data is not as skewed.)

EXAMPLE: Q2

Below are histograms of the accuracy and computation time taken for varying classification parameters. We want to be able to aggregate the two scores in order to be able to determine the best overall classifier.





Suggest some transformations that would be able to transform the variables so that it makes sense to take their average.

Example: Q2 Solution

Both of the variables are reasonably symmetric in terms of their distribution. For computation time, faster is better, so here we might use a negation and scale the data to the unit interval. Accuracy is already given over the scale 0 to 1. We could scale it so that a score of 80% corresponds with something closer to zero if we wanted, or we could leave it as is. We can consider whether we would want a classifier with an accuracy of 0.8 and the fastest time to have an aggregated score of about 0.9 or of about 0.5. The following uses the latter.

Accuracy: Let the minimum be a (and the maximum is 1). Denoting the original accuracy score by x and the transformed score by x':

$$x' = \frac{x - a}{1 - a}$$

Computation time: Let the minimum be a and the maximum be b, the original computation time is x and the transformed score for computation time will be x':

$$x' = 1 - \frac{x - a}{b - a}$$

EXAMPLES

1. If $PM(9, 10, 17, 16) = 13$ for some value of p, can we work out the value of $PM(12, 13, 20, 19)$ and $PM(18, 20, 34, 32)$ without knowing p?
2. If $p = -4$, what can we determine about the value of $PM(3, 0, 2, 8)$?
3. If $PM(3, 7, 29, 45) = 7.162$, is the value of p likely to be greater than 1 or less than 1?
4. If $PM(3, 7, 29, 45) = 36.258$, is the value of p likely to be greater than 1 or less than 1?
5. If $PM(2, 3, 8, 3) = 7$, is the value of p likely to be high or low?

EXAMPLES-SOLUTIONS

1. We can only determine $PM(12, 13, 20, 19)$ if we know p. If $p = 1$ then translation invariance will mean that our output should be 16 (the arithmetic mean), however if $p \neq 1$ then this will not hold. In this case, we can tell that p is 1 since the arithmetic mean of 9,10,17 and 16 should be 13. For $PM(18, 20, 34, 32)$, since the power mean is homogeneous for all p, we can deduce that the result should be 26 (double 13 since all the inputs are doubled).
2. The output will be zero since $x_2 = 0$ and for negative values of p, the power mean has 0 as an absorbent element.
3. This value of 7.162 seems closer to the two low inputs. Since the output would be 21 for $p = 1$, we can determine that p must be much lower.
4. This value is now closer to the higher inputs. We can tell that p must hence be higher than 1.
5. Since 7 is only relatively close to the 8, this power mean tends towards the highest input. That means that p must be high (it's close to 10).

Generating random samples

We can generate random samples with the **sample()** function.

1. First, generate random samples from `1` to `10`:

```
> sample(10)
```

2. If you would like to reproduce the same samples, you can set the random seed beforehand:

```
> set.seed(123)
> sample(10)
 [1]  3  8  4  7  6  1 10  9  2  5
```

3. You can then randomly choose two samples from `1` to `10`:

```
> sample(10,2)
[1] 10  5
```

4. If the population and sample size are required arguments, you can also use the `sample.int` function:

```
> sample.int(10,size=2)
[1] 7 6
```

Generate a sample from uniform distribution

1. First, we can create samples from uniform distribution by using the `runif` function:

```
> set.seed(123)
> uniform <- runif(n = 1000, min = 0, max = 1)
```

2. We can then make a histogram plot out of the samples from the uniform distribution:

```
> hist(uniform, main = "1,000 random variates from a uniform
distribution")
```

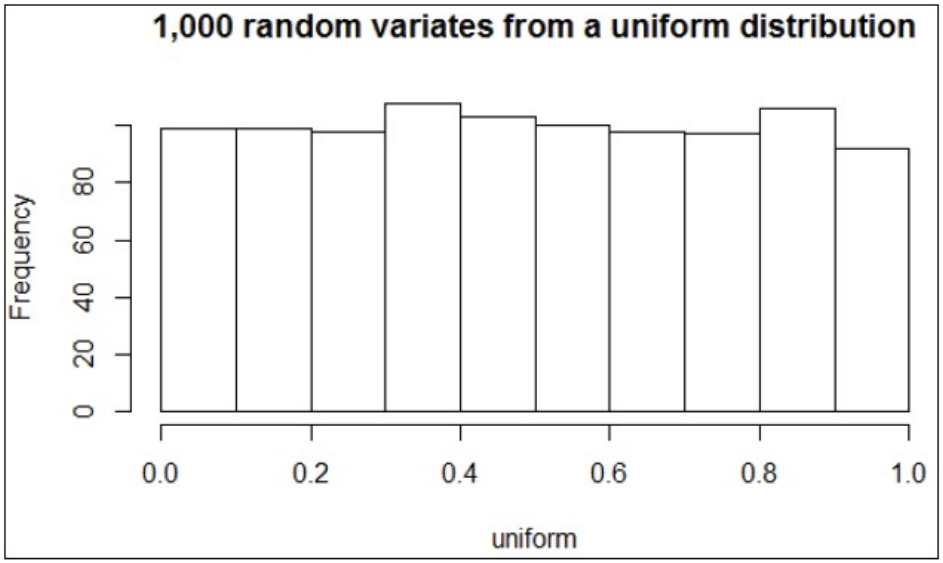


Figure 1: A histogram of 1,000 random variates from a uniform distribution

Sampling from a normal distribution

1. First, you can use the `rnorm` function to generate `30` and `1000` samples from a normal distribution:

```
> set.seed(123)
> data1 <- rnorm(30)
> data2 <- rnorm(1000)
```

2. We can then use the `hist` function to plot the histogram of `data1` and `data2`:

```
> par(mfrow = c(1,2))
> hist(data1, main="30 samples")
> hist(data2, main="1000 samples ")
```

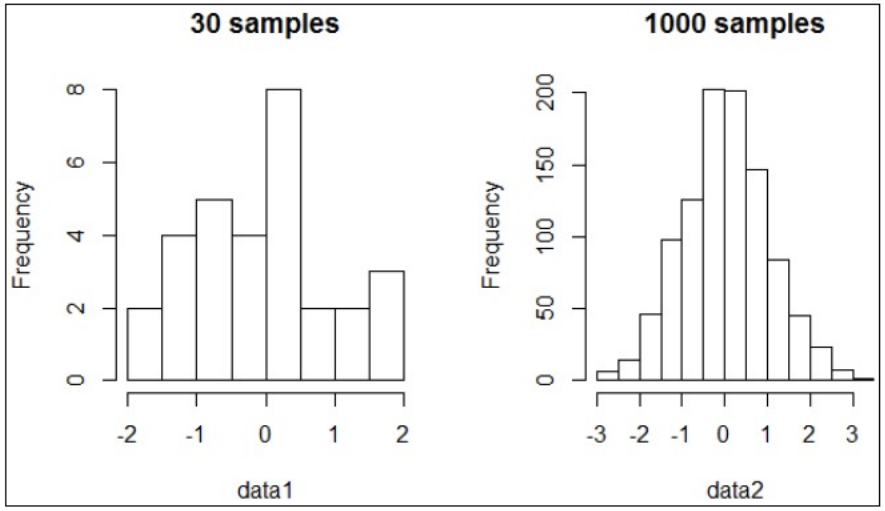


Figure 8: Histogram of 30 and 1,000 samples from a normal distribution

Sampling from a normal distribution (cont.)

3. To get the height of probability distribution at `0`, one can use the `dnorm` function:

```
> dnorm(0)
[1] 0.3989423
```

4. Moreover, to compute the area under the curve between *x* from `-1` to `1`, one can use the cumulative distribution function:

```
> pnorm(1) - pnorm(-1)
[1] 0.6826895
```

To plot the graph of `pnorm`, a `curve` function can be used:

```
> curve(pnorm(x), -3,3, main="Cumulative distribution function")
```

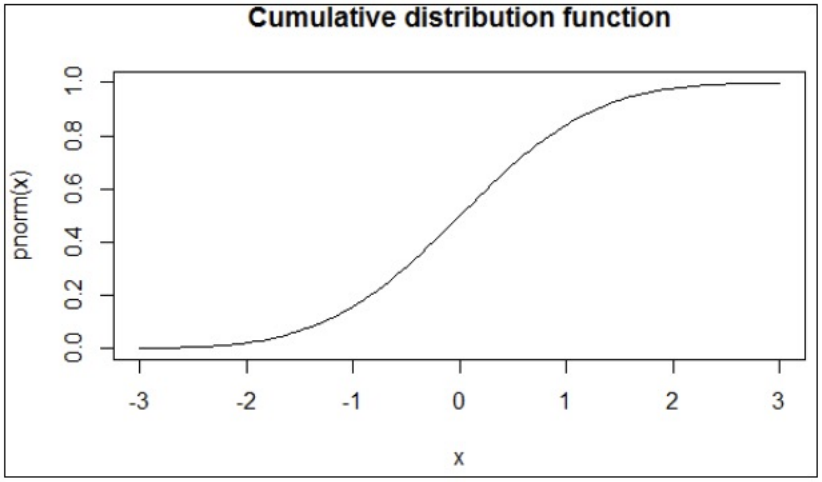


Figure 13: Cumulative distribution function

Generate binomial random variates

1. First, we can use `rbinom` to determine the frequency of drawing a six by rolling a dice 10 times:

```
> set.seed(123)
> rbinom(1, 10, 1/6)
[1] 1
```

2. Next, we can simulate 100 gamblers rolling a dice 10 times, and observe how many times a six is drawn by each gambler:

```
> set.seed(123)
> sim <- rbinom(100,10,1/6)
> table(sim)
sim
 0  1  2  3  4  5
17 36 23 18  4  2
```

```
> sim
  [1] 1 3 1 3 4 0 2 3 2 1 4 1 2 2 0 3 1 0 1 4 3 2 2 5 2 2 2 2 1 0 4 3 2 3 0 1 2 1 1 1 0 1 1 1 0 0 1 1 1 3
 [51] 0 1 3 0 2 1 0 2 3 1 2 0 1 1 3 1 3 3 3 1 2 2 2 0 1 1 1 2 1 0 1 2 1 3 0 1 5 3 3 1 0 2 1 2 1 1 3 0 1 2
```

Generate binomial random variates (cont.)

3. Additionally, we can simulate 1,000 people tossing a coin 10 times, and compute the number of heads at each tossing:

```
> set.seed(123)
> sim2 <- rbinom(1000,10,1/2)
> barplot(table(sim2), main="A simulation of 1,000 people tossing
a coin 10 times")
```

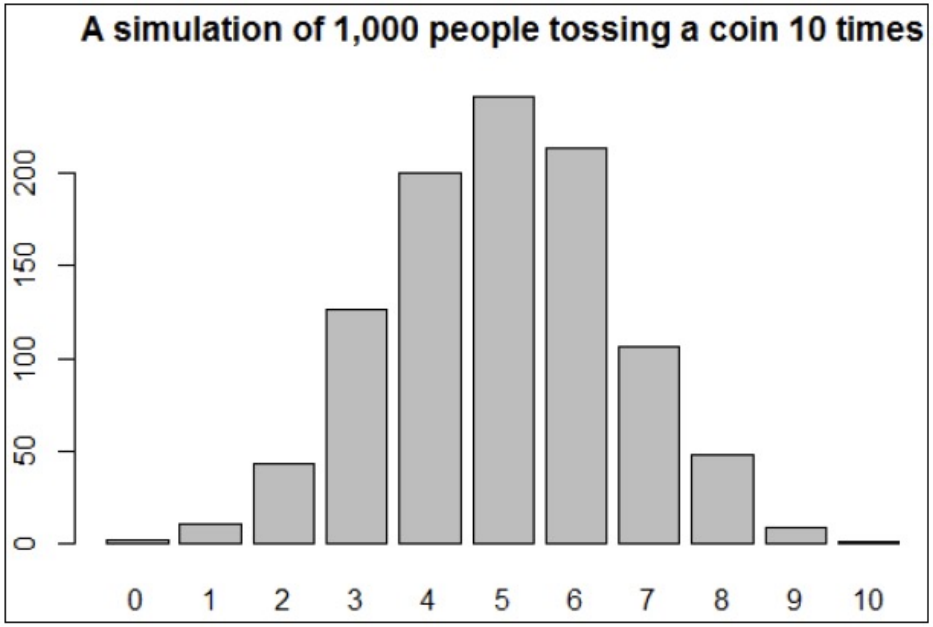


Figure 3: A simulation of 1,000 people tossing a coin 10 times

Generate binomial random variates (cont.)

4. We can then use `dbinom` to retrieve the probability of getting exactly one 6 when rolling a dice 10 times:

```
> dbinom(1, 10, 1/6)
[1] 0.3230112
```

5. If we would like to calculate the probability of getting less than three 6s from 10 dice rolls, we can use the `pbinom` function:

```
> dbinom(0, 10, 1/6) + dbinom(1, 10, 1/6) + dbinom(2, 10, 1/6) +
dbinom(3, 10, 1/6)
[1] 0.9302722
> pbinom(3, 10, 1/6)
[1] 0.9302722
```

Generating Poisson random variates

1. Similar to normal distribution, we can use `rpois` to generate samples from Poisson distribution:

```
> set.seed(123)
> poisson <- rpois(1000, lambda=3)
```

2. You can then plot sample data from a Poisson distribution into a histogram:

```
> hist(poisson, main="A histogram of a Poisson distribution")
```

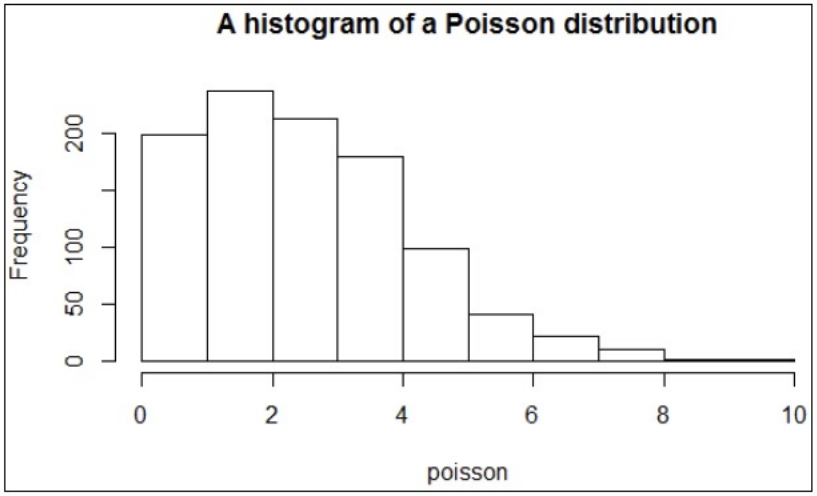


Figure 5: A histogram of a Poisson distribution

Generating Poisson random variates (cont.)

3. You can then obtain the height of the distribution function at $x=2$.

```
> dpois(2,lambda =3)
[1] 0.2240418
```

4. Here, we can use `barplot` to draw samples generated from the `dpois` function:

```
> barplot(dpois(0:10,lambda=3), names = 0:10, main="A barplot of a
Poisson distribution")
```

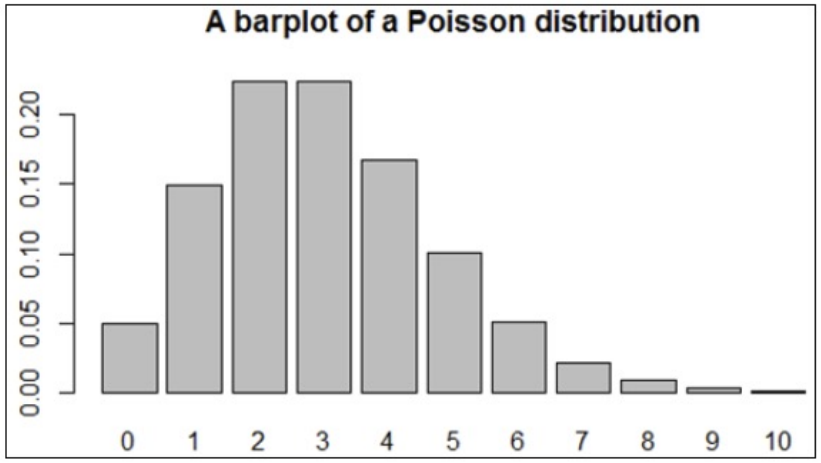


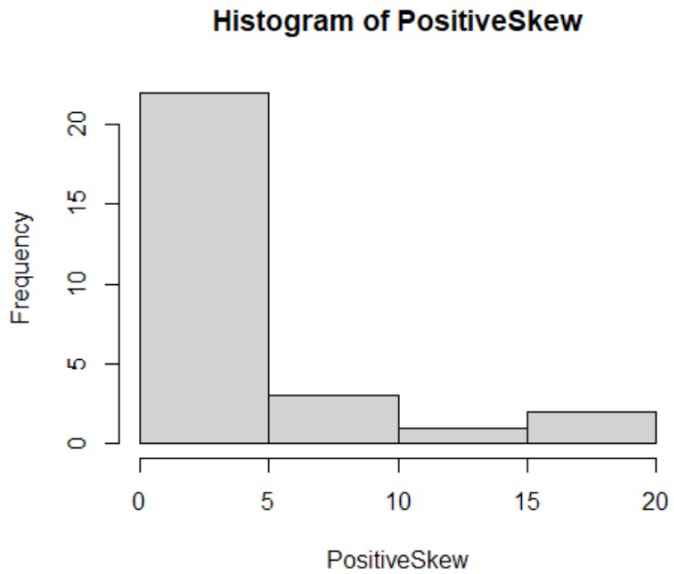
Figure 6: A bar plot of a Poisson distribution

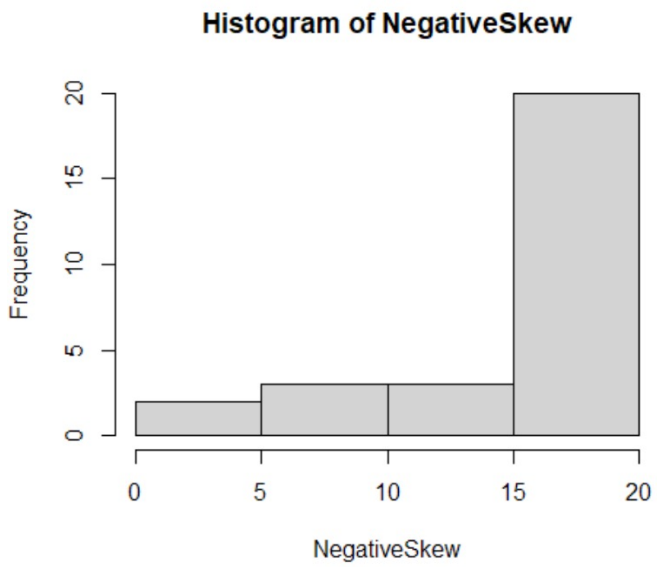
Transforming data with R

For right-skewed data—tail is on the right, positive skew—, common transformations include square root, cube root, and log.

For left-skewed data—tail is on the left, negative skew—, common transformations include second power, third power.

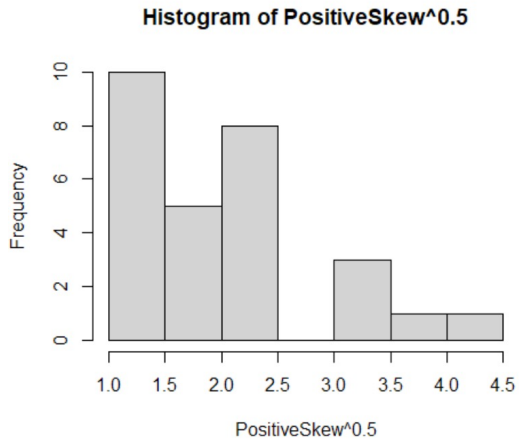
```
> PositiveSkew = c(1.0, 1.2, 1.1, 1.1, 2.4, 2.2, 2.6, 4.1, 5.0, 10.0, 4.0, 4.1, 4.2, 4.1, 5.1, 4.5,
5.0, 15.2, 10.0, 20.0, 1.1, 1.1, 1.2, 1.6, 2.2, 3.0, 4.0, 10.5)
> hist(PositiveSkew)
> NegativeSkew=20-PositiveSkew
> hist(NegativeSkew)
```



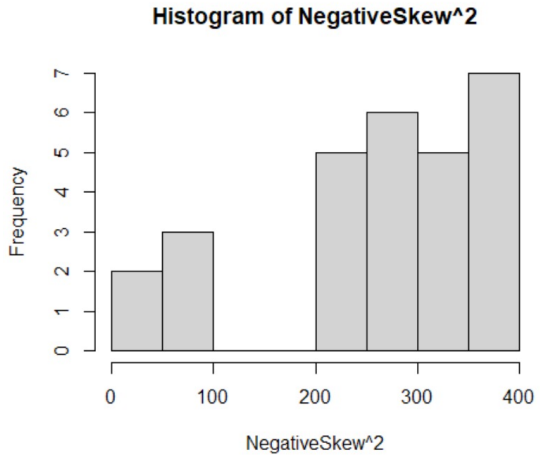


Transforming data with R (cont.)

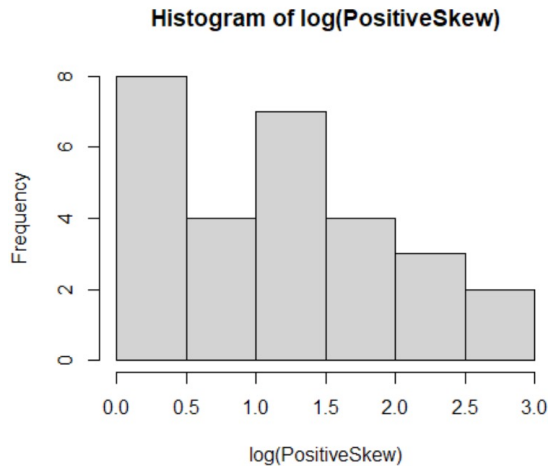
> hist(PositiveSkew^0.5)



> hist(NegativeSkew^2)



> hist(log(PositiveSkew))



> hist(NegativeSkew^3)