effectkeys.r

denis

2021-07-14

#!/usr/bin/r  
  
# We also refer to the arithmetic mean as just the “mean” because it is the most  
# commonly used mean.  
# It is often useful to think of the mean as an n-vector all of whose elements  
# are x̄. The symbol x̄ is also used to denote this vector; hence, we have  
mean(0:999, n = 1)

## [1] 499.5

# in which x̄ on the left-hand side is a vector and x̄ on the right-hand side  
# is a scalar. We also have, for the two diﬀerent objects,  
scale(0:7, center = TRUE, scale = TRUE)

## [,1]  
## [1,] -1.4288690  
## [2,] -1.0206207  
## [3,] -0.6123724  
## [4,] -0.2041241  
## [5,] 0.2041241  
## [6,] 0.6123724  
## [7,] 1.0206207  
## [8,] 1.4288690  
## attr(,"scaled:center")  
## [1] 3.5  
## attr(,"scaled:scale")  
## [1] 2.44949

# The meaning, whether a scalar or a vector, is usually clear from the con-  
# text. In any event, an expression such as x − x̄ is unambiguous; the addition  
# (subtraction) has the same meaning whether x̄ is interpreted as a vector or a  
# scalar. (In some mathematical treatments of vectors, addition of a scalar to  
# a vector is not deﬁned, but here we are following the conventions of modern  
# computer languages.)  
scales::ContinuousRange

## <ContinuousRange> object generator  
## Inherits from: <Range>  
## Public:  
## train: function (x)   
## reset: function ()   
## clone: function (deep = FALSE)   
## Parent env: <environment: namespace:scales>  
## Locked objects: TRUE  
## Locked class: FALSE  
## Portable: TRUE

# 2.2 Cartesian Coordinates and Geometrical  
# Properties of Vectors  
# Points in a Cartesian geometry can be identiﬁed with vectors. Several deﬁ-  
# nitions and properties of vectors can be motivated by this geometric inter-  
# pretation. In this interpretation, vectors are directed line segments with a  
# common origin. The geometrical properties can be seen most easily in terms  
# of a Cartesian coordinate system, but the properties of vectors deﬁned in  
# terms of a Cartesian geometry have analogues in Euclidean geometry without  
# a coordinate system. In such a system, only length and direction are deﬁned,  
# and two vectors are considered to be the same vector if they have the same  
# length and direction. Generally, we will not assume that there is a “position”  
# associated with a vector.  
geometric <- vector(mode = "logical", length = 0L)  
geometric

## logical(0)

# 2.2.1 Cartesian Geometry  
# A Cartesian coordinate system in d dimensions is deﬁned by d unit vectors,  
# e i in equation (2.3), each with d elements. A unit vector is also called a  
# principal axis of the coordinate system. The set of unit vectors is   
# orthonormal.  
# (There is an implied number of elements of a unit vector that is inferred from  
# the context. Also parenthetically, we remark that the phrase “unit vector” is  
# sometimes used to refer to a vector the sum of whose squared elements is 1,  
# that is, whose length, in the Euclidean distance sense, is 1. As we mentioned  
# above, we refer to this latter type of vector as a “normalized vector”.)  
# The sum of all of the unit vectors is the one vector:  
unit <- vector(mode = "logical", length = 0L)  
unit

## logical(0)

# A point x with Cartesian coordinates (x 1 , . . . , x d ) is associated with a  
# vector from the origin to the point, that is, the vector (x 1 , . . . , x d ).   
# The vector  
# can be written as the linear combination  
vec.linear = c(5,1,1,1,5,4)  
vec.linear

## [1] 5 1 1 1 5 4

# or, equivalently, as  
equiv.vector <- c(5,1,1,1,5,4)  
equiv.vector

## [1] 5 1 1 1 5 4

# 2.2.2 Projections  
# The projection of the vector y onto the vector x is the vector  
vec.y <- c(5)  
vec.y

## [1] 5

# This deﬁnition is consistent with a geometrical interpretation of vectors as  
# directed line segments with a common origin. The projection of y onto x is  
# the inner product of the normalized x and y times the normalized x; that is,  
# x̃, y x̃, where x̃ . Notice that the order of y and x is the same  
origin <- c(5, 5)  
origin

## [1] 5 5

# then r and ŷ are orthogonal, as we can easily see by taking their inner   
# product (see Figure 2.1). Notice also that the Pythagorean relationship holds:  
pythogorean <- c(2.1)  
pythogorean

## [1] 2.1

# As we mentioned on page 24, the mean ȳ can be interpreted either as a  
# scalar or as a vector all of whose elements are ȳ. As a vector, it is the   
# projection of y onto the one vector 1 n ,  
match.arg(arg = NULL, choices = c(0:535), several.ok = FALSE)

## [1] 0

# from equations (2.26) and (2.29).  
# We will consider more general projections (that is, projections onto planes  
# or other subspaces) on page 280, and on page 331 we will view linear   
# regression ﬁtting as a projection onto the space spanned by the independent   
# variables.  
page(535)  
  
# 2.2.3 Angles between Vectors  
# The angle between the vectors x and y is determined by its cosine, which we  
# can compute from the length of the projection of one vector onto the other.  
# Hence, denoting the angle between x and y as angle(x, y), we deﬁne  
angle <- c(5, 5)  
angle

## [1] 5 5

# with cos −1 (·) being taken in the interval [0, π]. The cosine is ±, with  
# the sign chosen appropriately; see Figure 2.1. Because of this choice of   
# cos −1 (·), we have that angle(y, x) = angle(x, y) — but see Exercise 2.13e   
# on page 39.  
n = 1  
cos(-1) + angle

## [1] 5.540302 5.540302

# Notice that the angle between two vectors is invariant to scaling of the  
# vectors; that is, for any nonzero scalar a, angle(ax, y) = angle(x, y).  
angle = angle  
angle

## [1] 5 5

# These quantities are called the direction cosines of the vector.  
# Although geometrical intuition often helps us in understanding properties  
# of vectors, sometimes it may lead us astray in high dimensions. Consider the  
# direction cosines of an arbitrary vector in a vector space with large   
# dimensions.  
# If the elements of the arbitrary vector are nearly equal (that is, if the   
# vector a diagonal through an orthant of the coordinate system), the direction   
# cosine goes to 0 as the dimension increases. In high dimensions, any two   
# vectors are “almost orthogonal” to each other; see Exercise 2.11.  
almost <- c("orthogonal", 1000)  
almost

## [1] "orthogonal" "1000"

# The geometric property of the angle between vectors has important im-  
# plications for certain operations both because it may indicate that rounding  
# in computations will have deleterious eﬀects and because it may indicate a  
# deﬁciency in the understanding of the application.  
library(effects)

## Loading required package: carData

## lattice theme set by effectsTheme()  
## See ?effectsTheme for details.

Effect.gls <- function(focal.predictors, mod, ...){  
 cl <- mod$call  
 cl$weights <- NULL  
 args <- list(  
 type = "glm",  
 call = cl,  
 formula = formula(mod),  
 family = NULL,  
 coefficients = coef(mod),  
 vcov = as.matrix(vcov(mod)),  
 method=NULL)  
 Effect.default(focal.predictors, mod, ..., sources=args)  
}  
  
require(rlme)

## Loading required package: rlme

g <- c(Employed ~ GNP + Population,  
 correlation=c(form= ~ Year), data=longley)  
format.default(g)

##   
## "Employed ~ GNP + Population"   
## correlation.form   
## "~Year"   
## data.GNP.deflator   
## "83.0, 88.5, 88.2, 89.5, 96.2, 98.1, 99.0, 100.0, 101.2, 104.6, 108.4, 110.8, 112.6, 114.2, 115.7, 116.9"   
## data.GNP   
## "234.289, 259.426, 258.054, 284.599, 328.975, 346.999, 365.385, 363.112, 397.469, 419.180, 442.769, 444.546, 482.704, 502.601, 518.173, 554.894"   
## data.Unemployed   
## "235.6, 232.5, 368.2, 335.1, 209.9, 193.2, 187.0, 357.8, 290.4, 282.2, 293.6, 468.1, 381.3, 393.1, 480.6, 400.7"   
## data.Armed.Forces   
## "159.0, 145.6, 161.6, 165.0, 309.9, 359.4, 354.7, 335.0, 304.8, 285.7, 279.8, 263.7, 255.2, 251.4, 257.2, 282.7"   
## data.Population   
## "107.608, 108.632, 109.773, 110.929, 112.075, 113.270, 115.094, 116.219, 117.388, 118.734, 120.445, 121.950, 123.366, 125.368, 127.852, 130.081"   
## data.Year   
## "1947, 1948, 1949, 1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957, 1958, 1959, 1960, 1961, 1962"   
## data.Employed   
## "60.323, 61.122, 60.171, 61.187, 63.221, 63.639, 64.989, 63.761, 66.019, 67.857, 68.169, 66.513, 68.655, 69.564, 69.331, 70.551"

print(g)

## [[1]]  
## Employed ~ GNP + Population  
##   
## $correlation.form  
## ~Year  
##   
## $data.GNP.deflator  
## [1] 83.0 88.5 88.2 89.5 96.2 98.1 99.0 100.0 101.2 104.6 108.4 110.8  
## [13] 112.6 114.2 115.7 116.9  
##   
## $data.GNP  
## [1] 234.289 259.426 258.054 284.599 328.975 346.999 365.385 363.112 397.469  
## [10] 419.180 442.769 444.546 482.704 502.601 518.173 554.894  
##   
## $data.Unemployed  
## [1] 235.6 232.5 368.2 335.1 209.9 193.2 187.0 357.8 290.4 282.2 293.6 468.1  
## [13] 381.3 393.1 480.6 400.7  
##   
## $data.Armed.Forces  
## [1] 159.0 145.6 161.6 165.0 309.9 359.4 354.7 335.0 304.8 285.7 279.8 263.7  
## [13] 255.2 251.4 257.2 282.7  
##   
## $data.Population  
## [1] 107.608 108.632 109.773 110.929 112.075 113.270 115.094 116.219 117.388  
## [10] 118.734 120.445 121.950 123.366 125.368 127.852 130.081  
##   
## $data.Year  
## [1] 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961  
## [16] 1962  
##   
## $data.Employed  
## [1] 60.323 61.122 60.171 61.187 63.221 63.639 64.989 63.761 66.019 67.857  
## [11] 68.169 66.513 68.655 69.564 69.331 70.551