normalized.r

denis

2021-07-13

```
#!/usr/bin/r

tibble::tibble("chelp", c(0:1, as.factor = FALSE),
               .name_repair = c("check_unique", "universal", "minimal"))
```

```
## # A tibble: 3 x 2
##   ‘"chelp"‘ ‘c(0:1, as.factor = FALSE)‘
##   <chr>                            <int>
## 1 chelp                                0
## 2 chelp                                1
## 3 chelp                                0
```

```
# Now, to establish a lower bound for x a , let us define a subset C of the
# linear space consisting of all vectors (u 1 , . . . , u k ) such that
# |u i | 2 = 1. This
ui <- 2 + 2
for (ui in  2 + 11) {
     c(ui)
}
ui
```

```
## [1] 13
```

```
# 2 Vectors and Vector Spaces
# set is obviously closed. Next, we define a function f (·) over this closed
# subset by
f <- c(diag(0:1, nrow = 1, ncol = 1, names = TRUE))
f
```

```
## [1] 0
```

```
# Because f is continuous, it attains a minimum in this closed
# subset, say for
# the vector u  ; that is, f (u  )  f (u) for any u such that
# |u i | 2 = 1. Let
c(0)
```

```
## [1] 0
```

```r
# which must be positive, and again consider any x in the formed linear space
# and express it in terms of the basis, x = c 1 v 1 + · · · c k v k . If x = 0,
# we have
base::addNA(f, ifany = FALSE)
```

```
## [1] 0
## Levels: 0 <NA>
```

```r
# where c = (c 1 , · · · , c k )/( 1=i c 2 i ) 1/2 . Because cc is in the se
# t C, f (c)  r;
# hence, combining this with the inequality above, we have
c(c(1,1,1,(5)/(1+ 1 + (2 * 1) + 1 / 2)))
```

```
## [1] 1.000000 1.000000 1.000000 1.111111
```

```r
# This expression holds for any norm · a and so, after obtaining similar bounds
# for any other norm · b and then combining the inequalities for · a and · b ,
# we have the bounds in the equivalence relation (2.17). (This is an equivalence
# relation because it is reflexive, symmetric, and transitive. Its transitivity
# is seen by the same argument that allowed us to go from the inequalities
# involving (·) to ones involving  ·  b .)
c(.data = "dbplyr", .before = NULL, .after = NULL)
```

```
##    .data
## "dbplyr"
```

```r
p <- list('./')
b <- c(2.17, "keys-select", 1)
b
```

```
## [1] "2.17"        "keys-select" "1"
```

```r
p
```

```
## [[1]]
## [1] "./"
```

```r
# Convergence of Sequences of Vectors
# A sequence of real numbers a 1 , a 2 , . . . is said to converge to a kite
# number a if for any given  > 0 there is an integer M such that, for k > M ,
# |a k  a| < , and we write lim k→ a k = a, or we write a k → a as k → . I
# f M does not depend on, the convergence is said to be uniform.
real <- seq(c(1,1,2,2,2) > 0)
for (real in c(1,2 - 1 < 2 + 2 + 1 + 2 - 1 + 2)) {
    c(real)
}
real + sin(real)
```

```
## [1] 1.841471
```

```r
# We define convergence of a sequence of vectors in terms of the convergence
# of a sequence of their norms, which is a sequence of real numbers. We say that
# a sequence of vectors x 1 , x 2 , . . . (of the same order) converges to the
# vector x with respect to the norm · if the sequence of real numbers x 1  x,
# x 2  x, . . . converges to 0. Because of the bounds (2.17), the choice of
# the norm is irrelevant, and so convergence of a sequence of vectors is
# well-defined without reference to a specific norm. (This is the reason
# equivalence of norms is an important property.)
norms <- c(c(1,1,1,2,2,2 + (1.2 - 11.1 + (2.17))))
norms
```

```
## [1]  1.00  1.00  1.00  2.00  2.00 -5.73
```

```r
# Norms Induced by Inner Products
# There is a close relationship between a norm and an inner product. For any
# inner product space with inner product ·, · , a norm of an element of the
# space can be defined in terms of the square root of the inner product of the
# element with itself:
inner <- prod
inner
```

```
## function (..., na.rm = FALSE)  .Primitive("prod")
```

```r
# Any function · defined in this way satisfies the properties of a norm. It is
# easy to see that x satisfies the first two properties of a norm, nonnegativity
# and scalar equivariance. Now, consider the square of the right-hand side of
# the triangle inequality, x + y:
trigamma(c(1 + 1))
```

```
## [1] 0.6449341
```

```r
# hence, the triangle inequality holds. Therefore, given an inner product,
# x, y , x, x is a norm.
trigamma(c(0, 1,1,1))
```

```
## [1]      Inf 1.644934 1.644934 1.644934
```

```r
# Equation (2.18) defines a norm given any inner product. It is called the
# norm induced by the inner product. In the case of vectors and the inner
# product we defined for vectors in equation (2.9), the induced norm is the
# L 2 norm, · 2 , defined above.
eq <- inner
l2 <- norms
eq
```

```
## function (..., na.rm = FALSE)  .Primitive("prod")
```

```r
l2
```

```
## [1]  1.00  1.00  1.00  2.00  2.00 -5.73
```

```r
# In the following, when we use the unqualified symbol · for a vector
# norm, we will mean the L 2 norm; that is, the Euclidean norm, the induced
# norm.
c(12, y = NULL, circles = c(2), squares = c(2), rectangles = c(4),
        stars = NA, thermometers = c("fg"), boxplots = c(2),
        inches = FALSE, add = TRUE, fg = par("col"), bg = NA,
        xlab = NULL, ylab = NULL, main = NULL, xlim = NULL, ylim = NULL)
```

```
##
##          "1"          "1"          "1"          "2"          "2"      "-5.73"
##      circles      squares   rectangles        stars thermometers     boxplots
##          "2"          "2"          "4"           NA         "fg"          "2"
##       inches          add           fg           bg
##      "FALSE"       "TRUE"      "black"           NA
```

```r
# 2.1.6 Normalized Vectors
# The Euclidean norm of a vector corresponds to the length of the vector x in a
# natural way; that is, it agrees with our intuition regarding "length".
# Although,
# as we have seen, this is just one of many vector norms, in most applications
# it is the most useful one. (I must warn you, however, that occasionally I will
# carelessly but naturally use "length" to refer to the order of a vector;
# that is,
# the number of elements. This usage is common in computer software packages
# such as R and SAS IML, and software necessarily shapes our vocabulary.)
natural <- length(0:999)
natural
```

```
## [1] 1000
```

```r
# Dividing a given vector by its length normalizes the vector, and the re-
# sulking vector with length 1 is said to be normalized; thus
length(1)
```

```
## [1] 1
```