mcvis: Multi-collinearity Visualisation

Kevin Wang

Initiated on 2020 Aug 22, compiled on 2020 Sep 21

Table of Contents

# Introduction

What is the most popular statistical model for a statistician? We think most statisticians would say the linear regression model. Indeed, linear regression is both rich in theory and powerful in practice with many flexible extensions such as generalised linear models and linear mixed models.

But what about the model limitations? Multicollinearity is perhaps the most obvious limitation of regression models, which can cause the least squares regression to produce unreasonable coefficient estimates and standard errors.

# Multicollinearity high correlation!

So what is multicollinearity? Formally, it is defined as the phenomenon when a group of predictor variables are exactly or approximately linearly dependent. In other words, if one predictor variable can be approximately expressed as a linear combination of other predictor variables, then we have a situation of multicollinearity.

**It is a common misconception that multicollinearity equals to high correlation between variables!** Consider the following example where we generate normally distributed predictor variables. The first predictor is constructed as plus some random noise. Looking at the correlation matrix, we don’t necessarily see anything alarming with all correlation coefficient being below 0.6. This might give us the false sense of security that no multicollinearity exist when in fact, the correlation of and the sum is approximately 0.998, enough to trigger multicollinearity for regression!

set.seed(123)  
p = 5  
n = 50  
  
X = matrix(rnorm(n\*p), ncol = p)  
X[,1] = X[,2] + X[,3] + X[,4] + rnorm(n, 0, 0.1)  
  
round(cor(X), 2)

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 1.00 0.42 0.57 0.55 0.02  
## [2,] 0.42 1.00 -0.16 -0.16 -0.13  
## [3,] 0.57 -0.16 1.00 -0.01 -0.04  
## [4,] 0.55 -0.16 -0.01 1.00 0.20  
## [5,] 0.02 -0.13 -0.04 0.20 1.00

cor(X[,1], X[,2] + X[,3] + X[,4])

## [1] 0.9978913

If we blindly fit a linear regression model not knowing of the existence of multicollinearity, then two things can happen. One is with the coefficient estimate itself, where we can obtain an estimate of that is opposite in sign to how we generated our response variable. Another consequence is on the standard error of the coefficient estimate. Looking at the summary output of the “full model” (with all predictors fitted), we see that the standard error of is ten times larger than that of the “true model” with only fitted.

y = 1\*X[,1] + rnorm(n)  
full\_model = lm(y ~ X)  
summary(full\_model)

##   
## Call:  
## lm(formula = y ~ X)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.96452 -0.86745 0.03628 0.77898 1.82110   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 0.01477 0.16207 0.091 0.928  
## X1 -0.31782 1.66770 -0.191 0.850  
## X2 1.40674 1.66037 0.847 0.401  
## X3 1.28008 1.64276 0.779 0.440  
## X4 1.26840 1.68674 0.752 0.456  
## X5 0.06178 0.16676 0.370 0.713  
##   
## Residual standard error: 1.073 on 44 degrees of freedom  
## Multiple R-squared: 0.67, Adjusted R-squared: 0.6325   
## F-statistic: 17.87 on 5 and 44 DF, p-value: 1.217e-09

true\_model = lm(y ~ X[,1])  
summary(true\_model)

##   
## Call:  
## lm(formula = y ~ X[, 1])  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -2.05035 -0.72315 -0.02727 0.74011 2.03745   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.001924 0.147449 0.013 0.99   
## X[, 1] 0.997030 0.103169 9.664 7.66e-13 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.042 on 48 degrees of freedom  
## Multiple R-squared: 0.6605, Adjusted R-squared: 0.6535   
## F-statistic: 93.39 on 1 and 48 DF, p-value: 7.659e-13

# Classic ways of diagnosing multicollinearity

So why do these strange phenomena occur? In least squares regression, both the estimator and its variance depend on the precision matrix . However, when the model design matrix, , is numerically “close” to having linearly dependent columns, as we do have when multicollinearity occurs, the matrix is not numerically invertible. Hence, elements of the precision matrix, regression estimators and standard errors start to behave erratically from a numeric perspective.

One way to “diagnose” multicollinearity is to compute the eigenvalues of the matrix and examine the value , where and are the maximum eigenvalue and the minimum eigenvalue, respectively. This ratio is known as the “condition number” of the design matrix and a rule of thumb is that if a condition number is larger than 30, then this indicates a severe case of multicollinearity (see e.g. Belsley et al. (1980, Section 3.2)). In our data example, we do indeed have such a severe case of multicollinearity.

evalues = eigen(t(X) %\*% X)$values  
condition\_number = sqrt(evalues[1]/evalues[p])  
  
condition\_number

## [1] 35.51142

However, if we didn’t simulate the data above and we obtained such a large condition number, how would we know which variables are the culprits, i.e. variables that contribute the most to causing multicollinearity? Afterall, the condition number only tells us of the existence of multicollinearity, but not which predictors are causing this. Knowning this information may help us to decide if any variables should be dropped or alternative modelling strategies might be needed (e.g. averaging these variables).

This is where mcvis can help!

# Introducting mcvis

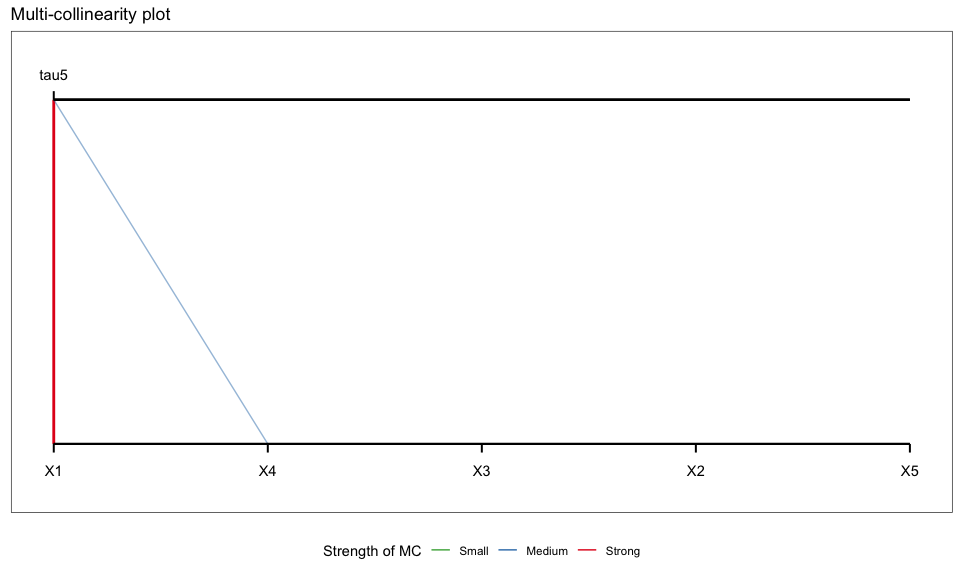
mcvis is a new R package that computes a new statistic called the MC-index which can diagnose multicollinearity. The theory behind this MC-index is published in Lin et. al. (2020). Briefly, MC-index is a vector of length matching that of the number of columns of . Each element of the MC-index vector is a number between zero and one, connecting the number , which measures ’s proximity to linear dependence, with a regression predictor variable. A larger MC-index value indicates that a variable is causing more multicollinearity than another variable. The computation of the MC-index uses a bootstrap resampling scheme to stabilise estimations and the result can be performed through the mcvis function in the mcvis package.

library(mcvis)  
mcvis\_result = mcvis(X)  
print(mcvis\_result)

## X1 X2 X3 X4 X5  
## tau5 0.51 0.12 0.15 0.22 0

In the output above, we see that , the first column of , is identified as the main variable causing multicollinearity. We can further visualise this via a (bipartite) graph that shows this connection between and the predictor variables. The size and colour of the lines are categorised by the magnitude of the MC-index value. This plot gives us a quick indication of the sources of the multicollinearity, ordered by the magnitude.

plot(mcvis\_result)



# Breast cancer

We will illustrate the use of the mcvis package using a real example from the dslabs data package. The brca data contains various important biopsy features for breast cancer cell nuclei. The original data measures features such as the radius for a collection of cell nuclei in a tumour and summarises these into mean, standard error and the worst value. The response variable is a factor with two levels denoting whether a tumour is malignant (“M”) or benign (“B”). To simplify interpretations, we will only focus on the first ten predictors, which correspond to the mean measurement of cell nuclei.

We do expect multicollinearity in this data because measures such as radius, perimeter and area of nucle should be highly correlated. Performing mcvis on this data, we do indeed see that this is the case and we will demonstrate that more than one group of variables are linearly highly correlated. We add some extra plotting parameters for mcvis to improve visibility.

library(dslabs)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(ggcorrplot)

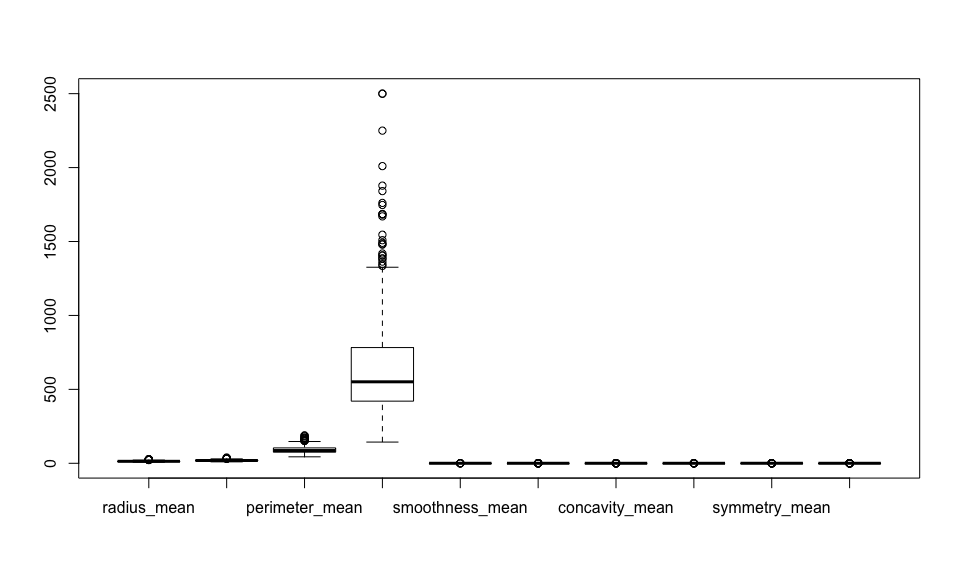
## Loading required package: ggplot2

data(brca)  
  
colnames(brca$x)

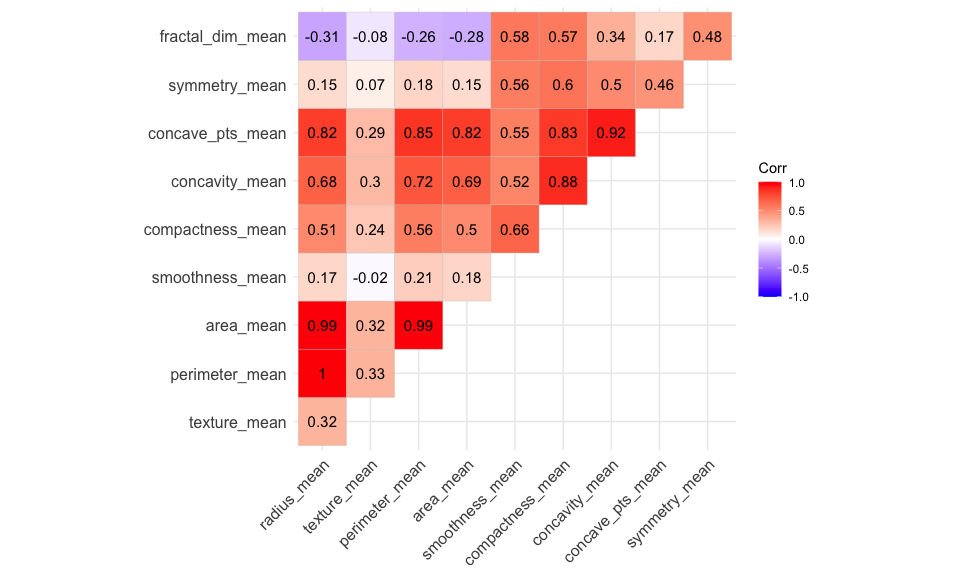
## [1] "radius\_mean" "texture\_mean" "perimeter\_mean"   
## [4] "area\_mean" "smoothness\_mean" "compactness\_mean"   
## [7] "concavity\_mean" "concave\_pts\_mean" "symmetry\_mean"   
## [10] "fractal\_dim\_mean" "radius\_se" "texture\_se"   
## [13] "perimeter\_se" "area\_se" "smoothness\_se"   
## [16] "compactness\_se" "concavity\_se" "concave\_pts\_se"   
## [19] "symmetry\_se" "fractal\_dim\_se" "radius\_worst"   
## [22] "texture\_worst" "perimeter\_worst" "area\_worst"   
## [25] "smoothness\_worst" "compactness\_worst" "concavity\_worst"   
## [28] "concave\_pts\_worst" "symmetry\_worst" "fractal\_dim\_worst"

x = as.data.frame(brca$x[,1:10])

## Some graphical summary of the data  
boxplot(x)



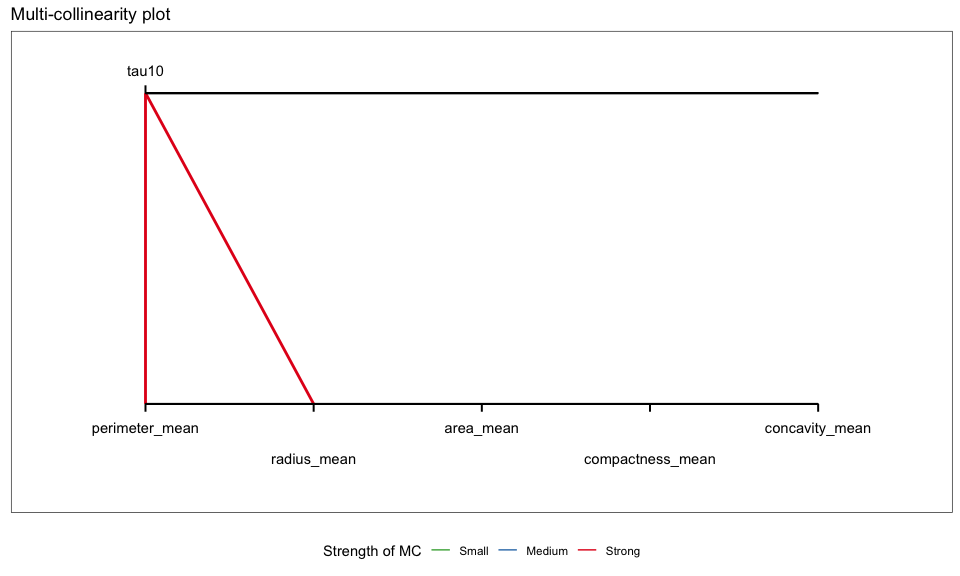
ggcorrplot(cor(x), type = "upper", lab = TRUE)



(mcvis\_result = mcvis::mcvis(x))

## radius\_mean texture\_mean perimeter\_mean area\_mean smoothness\_mean  
## tau10 0.36 0 0.64 0 0  
## compactness\_mean concavity\_mean concave\_pts\_mean symmetry\_mean  
## tau10 0 0 0 0  
## fractal\_dim\_mean  
## tau10 0

plot(mcvis\_result, var\_max = 5, label\_dodge = TRUE)

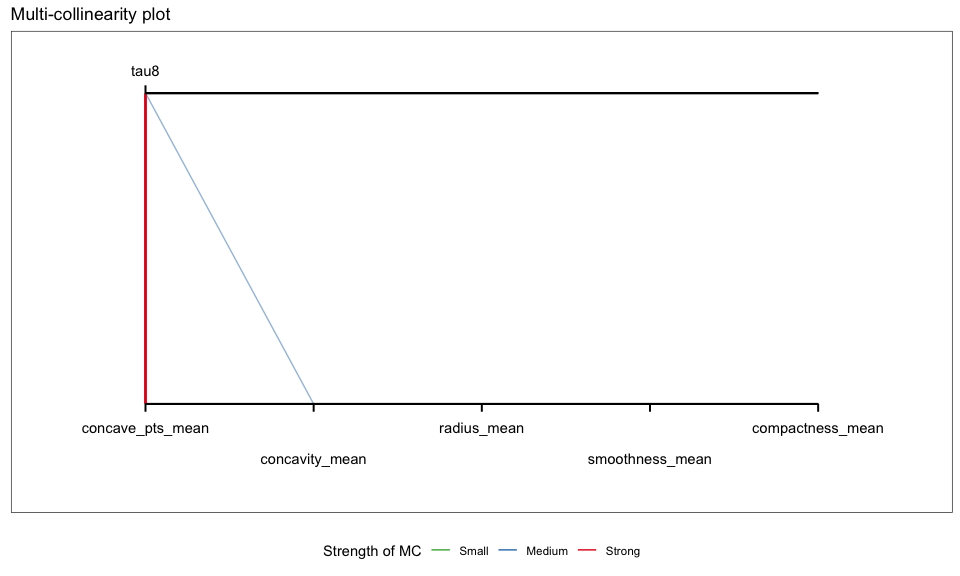


The MC-index identifies two variables as having a “Strong” MC-index value. Therefore, in this case we might be tempted to remove the perimeter\_mean and area\_mean from our design matrix as they provide very similar information as radius\_mean. Once the strongest sources of multicollinearity is addressed, additional groups of variables causing multicollinearity may be identified. We can repeat mcvis to identify concavity\_mean and concave\_pts\_mean as highly correlated variables.

x2 = dplyr::select(x, -perimeter\_mean, -area\_mean)  
  
(mcvis\_result2 = mcvis::mcvis(x2))

## radius\_mean texture\_mean smoothness\_mean compactness\_mean concavity\_mean  
## tau8 0.03 0 0.01 0 0.23  
## concave\_pts\_mean symmetry\_mean fractal\_dim\_mean  
## tau8 0.72 0 0

plot(mcvis\_result2, var\_max = 5, label\_dodge = TRUE)



Repeating mcvis again, we can also find that concave\_pts\_mean and concavity\_mean as another potential source of multicollinearity and as their definition are very similar according to the documentation, we may decide to remove one of these.

Of course, depending on data context, we might not want to simply remove the collinear variables like we have done here. Some common alternatives include averaging collinear predictors or using models that can decorrelate the predictor variables (e.g. random forest). Nonetheless, the diagnosis of multicollinearity is an important part of choosing which modelling strategy one should take.

# Conclusion

The linear regression model is arguably the most powerful tool in statistics. However, using it in practice may require extra checks and practical considerations with one often ignored issue being multicollinearity. mcvis is a tool that can help statisticians to identify different sources of multicollinearity to better decision making and the application of linear regression model.

# Reference

* Belsley, D. A., Kuh, E. & Welsch. R. E. Regression Diagnostics. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., 1980.
* Lin, C., Wang, K. & Mueller, S. mcvis: A new framework for collinearity discovery, diagnostic and visualization. Journal of Computational and Graphical Statistics In Press, (2020). DOI: 10.1080/10618600.2020.1779729

# Session Info

sessioninfo::session\_info()

## ─ Session info ───────────────────────────────────────────────────────────────  
## setting value   
## version R version 3.6.2 (2019-12-12)  
## os macOS Mojave 10.14.6   
## system x86\_64, darwin15.6.0   
## ui X11   
## language (EN)   
## collate en\_AU.UTF-8   
## ctype en\_AU.UTF-8   
## tz Australia/Melbourne   
## date 2020-09-21   
##   
## ─ Packages ───────────────────────────────────────────────────────────────────  
## package \* version date lib source   
## assertthat 0.2.1 2019-03-21 [1] CRAN (R 3.6.0)   
## cli 2.0.2 2020-02-28 [1] CRAN (R 3.6.0)   
## colorspace 1.4-1 2019-03-18 [1] CRAN (R 3.6.0)   
## crayon 1.3.4 2017-09-16 [1] CRAN (R 3.6.0)   
## digest 0.6.25 2020-02-23 [1] CRAN (R 3.6.0)   
## dplyr \* 1.0.0 2020-05-29 [1] CRAN (R 3.6.2)   
## dslabs \* 0.7.3 2019-12-13 [1] CRAN (R 3.6.0)   
## ellipsis 0.3.1 2020-05-15 [1] CRAN (R 3.6.2)   
## evaluate 0.14 2019-05-28 [1] CRAN (R 3.6.0)   
## fansi 0.4.1 2020-01-08 [1] CRAN (R 3.6.0)   
## farver 2.0.3 2020-01-16 [1] CRAN (R 3.6.0)   
## fastmap 1.0.1 2019-10-08 [1] CRAN (R 3.6.0)   
## generics 0.0.2 2018-11-29 [1] CRAN (R 3.6.0)   
## ggcorrplot \* 0.1.3 2019-05-19 [1] CRAN (R 3.6.0)   
## ggplot2 \* 3.3.2 2020-06-19 [1] CRAN (R 3.6.2)   
## glue 1.4.1 2020-05-13 [1] CRAN (R 3.6.2)   
## gtable 0.3.0 2019-03-25 [1] CRAN (R 3.6.0)   
## htmltools 0.5.0 2020-06-16 [1] CRAN (R 3.6.2)   
## httpuv 1.5.4 2020-06-06 [1] CRAN (R 3.6.2)   
## igraph 1.2.5 2020-03-19 [1] CRAN (R 3.6.0)   
## knitr 1.29 2020-06-23 [1] CRAN (R 3.6.2)   
## labeling 0.3 2014-08-23 [1] CRAN (R 3.6.0)   
## later 1.1.0.1 2020-06-05 [1] CRAN (R 3.6.2)   
## lattice 0.20-41 2020-04-02 [1] CRAN (R 3.6.2)   
## lifecycle 0.2.0 2020-03-06 [1] CRAN (R 3.6.0)   
## magrittr 1.5 2014-11-22 [1] CRAN (R 3.6.0)   
## mcvis \* 1.0.6 2020-09-21 [1] local   
## mime 0.9 2020-02-04 [1] CRAN (R 3.6.0)   
## mnormt 1.5-6 2020-02-03 [1] CRAN (R 3.6.0)   
## munsell 0.5.0 2018-06-12 [1] CRAN (R 3.6.0)   
## nlme 3.1-148 2020-05-24 [1] CRAN (R 3.6.2)   
## pillar 1.4.6 2020-07-10 [1] CRAN (R 3.6.2)   
## pkgconfig 2.0.3 2019-09-22 [1] CRAN (R 3.6.0)   
## plyr 1.8.6 2020-03-03 [1] CRAN (R 3.6.2)   
## promises 1.1.1 2020-06-09 [1] CRAN (R 3.6.2)   
## psych 1.9.12.31 2020-01-08 [1] CRAN (R 3.6.0)   
## purrr 0.3.4 2020-04-17 [1] CRAN (R 3.6.2)   
## R6 2.4.1 2019-11-12 [1] CRAN (R 3.6.0)   
## RColorBrewer 1.1-2 2014-12-07 [1] CRAN (R 3.6.0)   
## Rcpp 1.0.5 2020-07-06 [1] CRAN (R 3.6.2)   
## reshape2 1.4.4 2020-04-09 [1] CRAN (R 3.6.2)   
## rlang 0.4.7 2020-07-09 [1] CRAN (R 3.6.2)   
## rmarkdown 2.3.1 2020-06-23 [1] Github (rstudio/rmarkdown@b53a85a)  
## scales 1.1.1 2020-05-11 [1] CRAN (R 3.6.2)   
## sessioninfo 1.1.1 2018-11-05 [1] CRAN (R 3.6.0)   
## shiny 1.5.0 2020-06-23 [1] CRAN (R 3.6.2)   
## stringi 1.4.6 2020-02-17 [1] CRAN (R 3.6.1)   
## stringr 1.4.0 2019-02-10 [1] CRAN (R 3.6.0)   
## tibble 3.0.3 2020-07-10 [1] CRAN (R 3.6.2)   
## tidyselect 1.1.0 2020-05-11 [1] CRAN (R 3.6.2)   
## vctrs 0.3.1 2020-06-05 [1] CRAN (R 3.6.2)   
## withr 2.2.0 2020-04-20 [1] CRAN (R 3.6.2)   
## xfun 0.15 2020-06-21 [1] CRAN (R 3.6.2)   
## xtable 1.8-4 2019-04-21 [1] CRAN (R 3.6.0)   
## yaml 2.2.1 2020-02-01 [1] CRAN (R 3.6.1)   
##   
## [1] /Library/Frameworks/R.framework/Versions/3.6/Resources/library