**The story of corrr**

Moving to corrr - It started in decision making. My research data was responses to test batteries. My statistical bread and butter was regression-based techniques like multiple regression, path analysis, factor analysis (EFA and CFA), and structural equation modelling.

I spent a lot of time exploring correlation matrices to make model decisions, and diagnose poor fits or unexpected results!

**How to explore correlations?**

To illustrate some of the challenges I was facing, let's try explore some correlations with some very fancy data:

d <- mtcars

d$hp[3] <- NA

head(d)

#> mpg cyl disp hp drat wt qsec vs am gear carb

#> Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4

#> Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4

#> Datsun 710 22.8 4 108 NA 3.85 2.320 18.61 1 1 4 1

#> Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1

#> Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2

#> Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1

We could be motivated by [multicollinearity](https://en.wikipedia.org/wiki/Multicollinearity):

fit\_1 <- lm(mpg ~ hp, data = d)

fit\_2 <- lm(mpg ~ hp + disp, data = d)

summary(fit\_1)

#>

#> Call:

#> lm(formula = mpg ~ hp, data = d)

#>

#> Residuals:

#> Min 1Q Median 3Q Max

#> -5.7219 -2.2887 -0.8677 1.5815 8.1743

#>

#> Coefficients:

#> Estimate Std. Error t value Pr(>|t|)

#> (Intercept) 30.18457 1.69527 17.805 < 2e-16 \*\*\*

#> hp -0.06860 0.01039 -6.604 3.08e-07 \*\*\*

#> ---

#> Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

#>

#> Residual standard error: 3.925 on 29 degrees of freedom

#> (1 observation deleted due to missingness)

#> Multiple R-squared: 0.6006, Adjusted R-squared: 0.5868

#> F-statistic: 43.61 on 1 and 29 DF, p-value: 3.085e-07

summary(fit\_2)

#>

#> Call:

#> lm(formula = mpg ~ hp + disp, data = d)

#>

#> Residuals:

#> Min 1Q Median 3Q Max

#> -4.9136 -2.3472 -0.7302 1.9798 6.7549

#>

#> Coefficients:

#> Estimate Std. Error t value Pr(>|t|)

#> (Intercept) 30.963202 1.373067 22.550 < 2e-16 \*\*\*

#> hp -0.024840 0.013480 -1.843 0.075991 .

#> disp -0.030992 0.007504 -4.130 0.000296 \*\*\*

#> ---

#> Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

#>

#> Residual standard error: 3.149 on 28 degrees of freedom

#> (1 observation deleted due to missingness)

#> Multiple R-squared: 0.7518, Adjusted R-squared: 0.7341

#> F-statistic: 42.41 on 2 and 28 DF, p-value: 3.368e-09

Strange result. Let's check the correlations between mpg, hp, and disp to try and diagnose this problem. It should be simple using the base R function, cor(). Right?

Err, what is with all the NA's ?

rs <- cor(d)

rs

#> mpg cyl disp hp drat wt

#> mpg 1.0000000 -0.8521620 -0.8475514 NA 0.68117191 -0.8676594

#> cyl -0.8521620 1.0000000 0.9020329 NA -0.69993811 0.7824958

#> disp -0.8475514 0.9020329 1.0000000 NA -0.71021393 0.8879799

#> hp NA NA NA 1 NA NA

#> drat 0.6811719 -0.6999381 -0.7102139 NA 1.00000000 -0.7124406

#> wt -0.8676594 0.7824958 0.8879799 NA -0.71244065 1.0000000

#> qsec 0.4186840 -0.5912421 -0.4336979 NA 0.09120476 -0.1747159

#> vs 0.6640389 -0.8108118 -0.7104159 NA 0.44027846 -0.5549157

#> am 0.5998324 -0.5226070 -0.5912270 NA 0.71271113 -0.6924953

#> gear 0.4802848 -0.4926866 -0.5555692 NA 0.69961013 -0.5832870

#> carb -0.5509251 0.5269883 0.3949769 NA -0.09078980 0.4276059

#> qsec vs am gear carb

#> mpg 0.41868403 0.6640389 0.59983243 0.4802848 -0.55092507

#> cyl -0.59124207 -0.8108118 -0.52260705 -0.4926866 0.52698829

#> disp -0.43369788 -0.7104159 -0.59122704 -0.5555692 0.39497686

#> hp NA NA NA NA NA

#> drat 0.09120476 0.4402785 0.71271113 0.6996101 -0.09078980

#> wt -0.17471588 -0.5549157 -0.69249526 -0.5832870 0.42760594

#> qsec 1.00000000 0.7445354 -0.22986086 -0.2126822 -0.65624923

#> vs 0.74453544 1.0000000 0.16834512 0.2060233 -0.56960714

#> am -0.22986086 0.1683451 1.00000000 0.7940588 0.05753435

#> gear -0.21268223 0.2060233 0.79405876 1.0000000 0.27407284

#> carb -0.65624923 -0.5696071 0.05753435 0.2740728 1.00000000

Check the help page [?cor](https://stat.ethz.ch/R-manual/R-devel/library/stats/html/cor.html). Not so obvious. Default is use = "everything", and buried down in the details:

If use is "everything�?, NAs will propagate conceptually, i.e., a resulting value will be NA whenever one of its contributing observations is NA.

Have to handle missing values with use:

rs <- cor(d, use = "pairwise.complete.obs")

rs

#> mpg cyl disp hp drat wt

#> mpg 1.0000000 -0.8521620 -0.8475514 -0.7749843 0.68117191 -0.8676594

#> cyl -0.8521620 1.0000000 0.9020329 0.8298025 -0.69993811 0.7824958

#> disp -0.8475514 0.9020329 1.0000000 0.7860001 -0.71021393 0.8879799

#> hp -0.7749843 0.8298025 0.7860001 1.0000000 -0.44258306 0.6505927

#> drat 0.6811719 -0.6999381 -0.7102139 -0.4425831 1.00000000 -0.7124406

#> wt -0.8676594 0.7824958 0.8879799 0.6505927 -0.71244065 1.0000000

#> qsec 0.4186840 -0.5912421 -0.4336979 -0.7064809 0.09120476 -0.1747159

#> vs 0.6640389 -0.8108118 -0.7104159 -0.7162015 0.44027846 -0.5549157

#> am 0.5998324 -0.5226070 -0.5912270 -0.2196179 0.71271113 -0.6924953

#> gear 0.4802848 -0.4926866 -0.5555692 -0.1161968 0.69961013 -0.5832870

#> carb -0.5509251 0.5269883 0.3949769 0.7437843 -0.09078980 0.4276059

#> qsec vs am gear carb

#> mpg 0.41868403 0.6640389 0.59983243 0.4802848 -0.55092507

#> cyl -0.59124207 -0.8108118 -0.52260705 -0.4926866 0.52698829

#> disp -0.43369788 -0.7104159 -0.59122704 -0.5555692 0.39497686

#> hp -0.70648093 -0.7162015 -0.21961793 -0.1161968 0.74378428

#> drat 0.09120476 0.4402785 0.71271113 0.6996101 -0.09078980

#> wt -0.17471588 -0.5549157 -0.69249526 -0.5832870 0.42760594

#> qsec 1.00000000 0.7445354 -0.22986086 -0.2126822 -0.65624923

#> vs 0.74453544 1.0000000 0.16834512 0.2060233 -0.56960714

#> am -0.22986086 0.1683451 1.00000000 0.7940588 0.05753435

#> gear -0.21268223 0.2060233 0.79405876 1.0000000 0.27407284

#> carb -0.65624923 -0.5696071 0.05753435 0.2740728 1.00000000

Can we focus on subset with dplyr? Nope.

dplyr::select(rs, mpg, hp, disp)

#> Error in UseMethod("select\_"): no applicable method for 'select\_' applied to an object of class "c('matrix', 'double', 'numeric')"

Riiiiiight! It's a matrix and dplyr is for data frames.

class(rs)

#> [1] "matrix"

So we can use square brackets with matrices? Or not…

vars <- c("mpg", "hp", "disp")

rs[rownames(rs) %in% vars]

#> [1] 1.0000000 -0.8475514 -0.7749843 -0.8521620 0.9020329 0.8298025

#> [7] -0.8475514 1.0000000 0.7860001 -0.7749843 0.7860001 1.0000000

#> [13] 0.6811719 -0.7102139 -0.4425831 -0.8676594 0.8879799 0.6505927

#> [19] 0.4186840 -0.4336979 -0.7064809 0.6640389 -0.7104159 -0.7162015

#> [25] 0.5998324 -0.5912270 -0.2196179 0.4802848 -0.5555692 -0.1161968

#> [31] -0.5509251 0.3949769 0.7437843

Mm, square brackets can take on different functions with matrices. Without a comma, it's treated like a vector. With a comma, we can separately specify the dimensions.

vars <- c("mpg", "hp", "disp")

rs[rownames(rs) %in% vars, colnames(rs) %in% vars]

#> mpg disp hp

#> mpg 1.0000000 -0.8475514 -0.7749843

#> disp -0.8475514 1.0000000 0.7860001

#> hp -0.7749843 0.7860001 1.0000000

Aha! High correlation between input variables (multicollinearity).

But seriously, this syntax is pretty ugly.

vars <- c("mpg", "hp", "disp")

rs[rownames(rs) %in% vars, colnames(rs) %in% vars]

We diagnosed our multicollinearity problem. What if we want to something a bit more complex like exploring clustering of variables in high dimensional space? Could use exploratory factor analysis.

factanal(na.omit(d), factors = 2)

#>

#> Call:

#> factanal(x = na.omit(d), factors = 2)

#>

#> Uniquenesses:

#> mpg cyl disp hp drat wt qsec vs am gear carb

#> 0.155 0.072 0.101 0.145 0.295 0.174 0.154 0.264 0.160 0.245 0.387

#>

#> Loadings:

#> Factor1 Factor2

#> mpg 0.694 -0.603

#> cyl -0.629 0.729

#> disp -0.730 0.605

#> hp -0.335 0.862

#> drat 0.810 -0.221

#> wt -0.809 0.414

#> qsec -0.162 -0.905

#> vs 0.280 -0.811

#> am 0.910 0.110

#> gear 0.859 0.133

#> carb 0.781

#>

#> Factor1 Factor2

#> SS loadings 4.506 4.342

#> Proportion Var 0.410 0.395

#> Cumulative Var 0.410 0.804

#>

#> Test of the hypothesis that 2 factors are sufficient.

#> The chi square statistic is 68.49 on 34 degrees of freedom.

#> The p-value is 0.000414

factanal(na.omit(d), factors = 5)

#>

#> Call:

#> factanal(x = na.omit(d), factors = 5)

#>

#> Uniquenesses:

#> mpg cyl disp hp drat wt qsec vs am gear carb

#> 0.112 0.044 0.012 0.066 0.275 0.005 0.091 0.128 0.136 0.143 0.045

#>

#> Loadings:

#> Factor1 Factor2 Factor3 Factor4 Factor5

#> mpg 0.647 -0.441 -0.483 -0.203

#> cyl -0.604 0.694 0.293 0.128

#> disp -0.645 0.543 0.202 0.451 0.182

#> hp -0.264 0.659 0.544 0.205 0.305

#> drat 0.806 -0.250

#> wt -0.725 0.226 0.420 0.488

#> qsec -0.184 -0.890 -0.269

#> vs 0.242 -0.848 -0.209 -0.155 0.163

#> am 0.916 0.118

#> gear 0.892 0.237

#> carb 0.119 0.458 0.853

#>

#> Factor1 Factor2 Factor3 Factor4 Factor5

#> SS loadings 4.186 3.255 1.739 0.582 0.180

#> Proportion Var 0.381 0.296 0.158 0.053 0.016

#> Cumulative Var 0.381 0.676 0.835 0.887 0.904

#>

#> Test of the hypothesis that 5 factors are sufficient.

#> The chi square statistic is 2.17 on 10 degrees of freedom.

#> The p-value is 0.995

So many questions! I'd much rather explore the correlations.

Let's try to find all variables with a correlation greater than 0.90. Why doesn't this work?!

col\_has\_over\_90 <- apply(rs, 2, function(x) any(x > .9))

rs[, col\_has\_over\_90]

#> mpg cyl disp hp drat wt

#> mpg 1.0000000 -0.8521620 -0.8475514 -0.7749843 0.68117191 -0.8676594

#> cyl -0.8521620 1.0000000 0.9020329 0.8298025 -0.69993811 0.7824958

#> disp -0.8475514 0.9020329 1.0000000 0.7860001 -0.71021393 0.8879799

#> hp -0.7749843 0.8298025 0.7860001 1.0000000 -0.44258306 0.6505927

#> drat 0.6811719 -0.6999381 -0.7102139 -0.4425831 1.00000000 -0.7124406

#> wt -0.8676594 0.7824958 0.8879799 0.6505927 -0.71244065 1.0000000

#> qsec 0.4186840 -0.5912421 -0.4336979 -0.7064809 0.09120476 -0.1747159

#> vs 0.6640389 -0.8108118 -0.7104159 -0.7162015 0.44027846 -0.5549157

#> am 0.5998324 -0.5226070 -0.5912270 -0.2196179 0.71271113 -0.6924953

#> gear 0.4802848 -0.4926866 -0.5555692 -0.1161968 0.69961013 -0.5832870

#> carb -0.5509251 0.5269883 0.3949769 0.7437843 -0.09078980 0.4276059

#> qsec vs am gear carb

#> mpg 0.41868403 0.6640389 0.59983243 0.4802848 -0.55092507

#> cyl -0.59124207 -0.8108118 -0.52260705 -0.4926866 0.52698829

#> disp -0.43369788 -0.7104159 -0.59122704 -0.5555692 0.39497686

#> hp -0.70648093 -0.7162015 -0.21961793 -0.1161968 0.74378428

#> drat 0.09120476 0.4402785 0.71271113 0.6996101 -0.09078980

#> wt -0.17471588 -0.5549157 -0.69249526 -0.5832870 0.42760594

#> qsec 1.00000000 0.7445354 -0.22986086 -0.2126822 -0.65624923

#> vs 0.74453544 1.0000000 0.16834512 0.2060233 -0.56960714

#> am -0.22986086 0.1683451 1.00000000 0.7940588 0.05753435

#> gear -0.21268223 0.2060233 0.79405876 1.0000000 0.27407284

#> carb -0.65624923 -0.5696071 0.05753435 0.2740728 1.00000000

The diagonal is 1. All cols have a value greater than .90!

Exclude diagonal:

diag(rs) <- NA

col\_has\_over\_90 <- apply(rs, 2, function(x) any(x > .9, na.rm = TRUE))

rs[, col\_has\_over\_90]

#> cyl disp

#> mpg -0.8521620 -0.8475514

#> cyl NA 0.9020329

#> disp 0.9020329 NA

#> hp 0.8298025 0.7860001

#> drat -0.6999381 -0.7102139

#> wt 0.7824958 0.8879799

#> qsec -0.5912421 -0.4336979

#> vs -0.8108118 -0.7104159

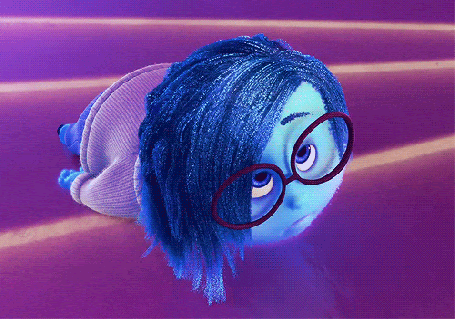
#> am -0.5226070 -0.5912270

#> gear -0.4926866 -0.5555692

#> carb 0.5269883 0.3949769

Again, this syntax is pretty gross. Imagine showing this to a beginner and asking them to write down as much as they remember. Probably not much would be my guess.

What about vizualising correlations? I'd suggest giving up at this point.

[](https://i1.wp.com/svbtleusercontent.com/vCAdmmx28nbzoKYcGVrh210xspap.gif?ssl=1)

**Exploring data with the tidyverse**

Remember me as postgrad? I'd discovered the tidyverse and really liked it, because *exploring* data with the tidyverse is easy.

library(tidyverse)

d %>%

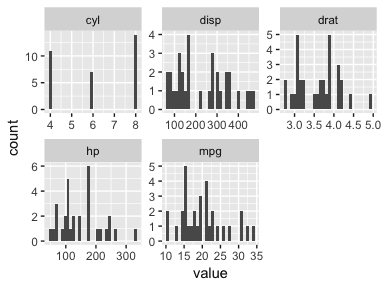
select(mpg:drat) %>%

gather() %>%

ggplot(aes(value)) +

geom\_histogram() +

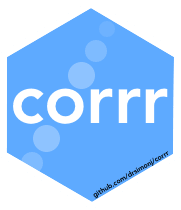
facet\_wrap(~key, scales = "free")

[](https://i2.wp.com/svbtleusercontent.com/v8a6SMH3qEUWDAnK2fbtr60xspap.png?ssl=1)

Can't we have this for correlations? I don't want to do any crazy mathematical operations or statistical tests. I just want to quickly explore the value. It's not a big ask.

Good news! This is why I developed corrr as a tidyverse-style package for exploring correlations in R.

**corrr**

[](https://i0.wp.com/svbtleusercontent.com/oDU6kAGErMQCpLiuDJoTRX0xspap.png?ssl=1)

Here's a quick example to get a feel for the syntax:

library(corrr)

d %>%

correlate() %>%

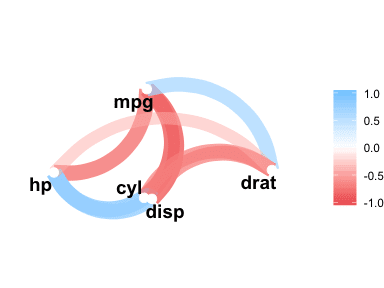
focus(mpg:drat, mirror = TRUE) %>%

network\_plot()

#>

#> Correlation method: 'pearson'

#> Missing treated using: 'pairwise.complete.obs'

[](https://i0.wp.com/svbtleusercontent.com/u5EHjmYYpN2zSdm5i3ZTri0xspap.png?ssl=1)

The first objective of corrr was to write a new function that uses cor() but:

* Handles missing by default with use = "pairwise.complete.obs"
* Stops diagonal from getting in the way by setting it to NA
* Makes it possible to use tidyverse tools by returning a data.frame instead of a matrix

Now, meet correlate()

rs <- correlate(d)

#>

#> Correlation method: 'pearson'

#> Missing treated using: 'pairwise.complete.obs'

rs

#> # A tibble: 11 x 12

#> rowname mpg cyl disp hp drat wt qsec

#>

#> 1 mpg NA -0.852 -0.848 -0.775 0.681 -0.868 0.419

#> 2 cyl -0.852 NA 0.902 0.830 -0.700 0.782 -0.591

#> 3 disp -0.848 0.902 NA 0.786 -0.710 0.888 -0.434

#> 4 hp -0.775 0.830 0.786 NA -0.443 0.651 -0.706

#> 5 drat 0.681 -0.700 -0.710 -0.443 NA -0.712 0.0912

#> 6 wt -0.868 0.782 0.888 0.651 -0.712 NA -0.175

#> 7 qsec 0.419 -0.591 -0.434 -0.706 0.0912 -0.175 NA

#> 8 vs 0.664 -0.811 -0.710 -0.716 0.440 -0.555 0.745

#> 9 am 0.600 -0.523 -0.591 -0.220 0.713 -0.692 -0.230

#> 10 gear 0.480 -0.493 -0.556 -0.116 0.700 -0.583 -0.213

#> 11 carb -0.551 0.527 0.395 0.744 -0.0908 0.428 -0.656

#> # ... with 4 more variables: vs , am , gear , carb

Same args as cor() with some extras

correlate(d, method = "spearman", diagonal = 1)

#>

#> Correlation method: 'spearman'

#> Missing treated using: 'pairwise.complete.obs'

#> # A tibble: 11 x 12

#> rowname mpg cyl disp hp drat wt qsec vs

#>

#> 1 mpg 1.00 -0.911 -0.909 -0.889 0.651 -0.886 0.467 0.707

#> 2 cyl -0.911 1.00 0.928 0.899 -0.679 0.858 -0.572 -0.814

#> 3 disp -0.909 0.928 1.00 0.845 -0.684 0.898 -0.460 -0.724

#> 4 hp -0.889 0.899 0.845 1.00 -0.504 0.763 -0.660 -0.747

#> 5 drat 0.651 -0.679 -0.684 -0.504 1.00 -0.750 0.0919 0.447

#> 6 wt -0.886 0.858 0.898 0.763 -0.750 1.00 -0.225 -0.587

#> 7 qsec 0.467 -0.572 -0.460 -0.660 0.0919 -0.225 1.00 0.792

#> 8 vs 0.707 -0.814 -0.724 -0.747 0.447 -0.587 0.792 1.00

#> 9 am 0.562 -0.522 -0.624 -0.326 0.687 -0.738 -0.203 0.168

#> 10 gear 0.543 -0.564 -0.594 -0.316 0.745 -0.676 -0.148 0.283

#> 11 carb -0.657 0.580 0.540 0.723 -0.125 0.500 -0.659 -0.634

#> # ... with 3 more variables: am , gear , carb

The output of correlate()

* A helpful message to remind us of what's happening (turned off with quiet = TRUE)
* A correlation data frame (tibble) with class cor\_df and:
  + Diagonals set to NA (adjusted via diagonal = NA)
  + A "rowname" colum rather than rownames
  + It's now super easy to pipe straight into tidyverse functions that work with data frames. For example:

rs %>%

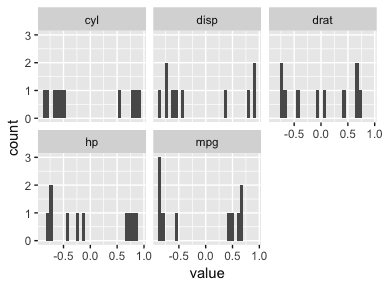
select(mpg:drat) %>%

gather() %>%

ggplot(aes(value)) +

geom\_histogram() +

facet\_wrap(~key)

[](https://i2.wp.com/svbtleusercontent.com/tJ2drcGTPLtBBpks6AkPBP0xspap.png?ssl=1)

How about that challenge to find cols with a correlation greater than .9?

any\_over\_90 <- function(x) any(x > .9, na.rm = TRUE)

rs %>% select\_if(any\_over\_90)

#> # A tibble: 11 x 3

#> rowname cyl disp

#>

#> 1 mpg -0.852 -0.848

#> 2 cyl NA 0.902

#> 3 disp 0.902 NA

#> 4 hp 0.830 0.786

#> 5 drat -0.700 -0.710

#> 6 wt 0.782 0.888

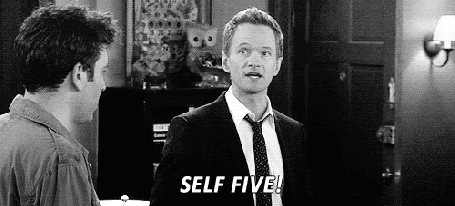
#> 7 qsec -0.591 -0.434

#> 8 vs -0.811 -0.710

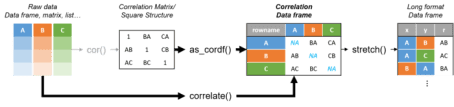
#> 9 am -0.523 -0.591

#> 10 gear -0.493 -0.556

#> 11 carb 0.527 0.395

[](https://i2.wp.com/svbtleusercontent.com/mf3pUQeQCFNio3s5ur5hJD0xspap.gif?ssl=1)

Here's a diagram to get you started after library(corrr):

[](https://i1.wp.com/svbtleusercontent.com/jCGEz9c5hotdLGYy9ewnHP0xspap.png?ssl=1)

**Correlation data frames are not tidy**

Tidy data functions target columns OR rows, but I found myself frequently wanting to make changes to both. So next came the ability to focus() on columns and rows. This function acts just like dplyr's select(), but also excludes the selected colums from the rows (or everything else with the mirror argument).

rs %>%

focus(mpg, disp, hp)

#> # A tibble: 8 x 4

#> rowname mpg disp hp

#>

#> 1 cyl -0.852 0.902 0.830

#> 2 drat 0.681 -0.710 -0.443

#> 3 wt -0.868 0.888 0.651

#> 4 qsec 0.419 -0.434 -0.706

#> 5 vs 0.664 -0.710 -0.716

#> 6 am 0.600 -0.591 -0.220

#> 7 gear 0.480 -0.556 -0.116

#> 8 carb -0.551 0.395 0.744

rs %>%

focus(-mpg, -disp, -hp)

#> # A tibble: 3 x 9

#> rowname cyl drat wt qsec vs am gear carb

#>

#> 1 mpg -0.852 0.681 -0.868 0.419 0.664 0.600 0.480 -0.551

#> 2 disp 0.902 -0.710 0.888 -0.434 -0.710 -0.591 -0.556 0.395

#> 3 hp 0.830 -0.443 0.651 -0.706 -0.716 -0.220 -0.116 0.744

rs %>%

focus(mpg, disp, hp, mirror = TRUE)

#> # A tibble: 3 x 4

#> rowname mpg disp hp

#>

#> 1 mpg NA -0.848 -0.775

#> 2 disp -0.848 NA 0.786

#> 3 hp -0.775 0.786 NA

rs %>%

focus(matches("^d"))

#> # A tibble: 9 x 3

#> rowname disp drat

#>

#> 1 mpg -0.848 0.681

#> 2 cyl 0.902 -0.700

#> 3 hp 0.786 -0.443

#> 4 wt 0.888 -0.712

#> 5 qsec -0.434 0.0912

#> 6 vs -0.710 0.440

#> 7 am -0.591 0.713

#> 8 gear -0.556 0.700

#> 9 carb 0.395 -0.0908

rs %>%

focus\_if(any\_over\_90, mirror = TRUE)

#> # A tibble: 2 x 3

#> rowname cyl disp

#>

#> 1 cyl NA 0.902

#> 2 disp 0.902 NA

One of my favourite uses is to focus() on correlations of one variable with all others and plot the results.

rs %>%

focus(mpg)

#> # A tibble: 10 x 2

#> rowname mpg

#>

#> 1 cyl -0.852

#> 2 disp -0.848

#> 3 hp -0.775

#> 4 drat 0.681

#> 5 wt -0.868

#> 6 qsec 0.419

#> 7 vs 0.664

#> 8 am 0.600

#> 9 gear 0.480

#> 10 carb -0.551

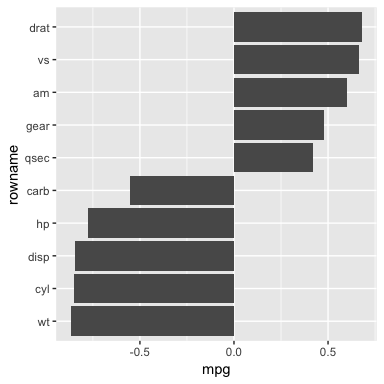
rs %>%

focus(mpg) %>%

mutate(rowname = reorder(rowname, mpg)) %>%

ggplot(aes(rowname, mpg)) +

geom\_col() + coord\_flip()

[](https://i0.wp.com/svbtleusercontent.com/iNTqBGjoFnZCbxPnRhHhF40xspap.png?ssl=1)

You can also rearrange() the entire data frame based on clustering algorithms:

rs %>% rearrange()

#> # A tibble: 11 x 12

#> rowname am gear drat wt disp mpg cyl

#>

#> 1 am NA 0.794 0.713 -0.692 -0.591 0.600 -0.523

#> 2 gear 0.794 NA 0.700 -0.583 -0.556 0.480 -0.493

#> 3 drat 0.713 0.700 NA -0.712 -0.710 0.681 -0.700

#> 4 wt -0.692 -0.583 -0.712 NA 0.888 -0.868 0.782

#> 5 disp -0.591 -0.556 -0.710 0.888 NA -0.848 0.902

#> 6 mpg 0.600 0.480 0.681 -0.868 -0.848 NA -0.852

#> 7 cyl -0.523 -0.493 -0.700 0.782 0.902 -0.852 NA

#> 8 vs 0.168 0.206 0.440 -0.555 -0.710 0.664 -0.811

#> 9 hp -0.220 -0.116 -0.443 0.651 0.786 -0.775 0.830

#> 10 carb 0.0575 0.274 -0.0908 0.428 0.395 -0.551 0.527

#> 11 qsec -0.230 -0.213 0.0912 -0.175 -0.434 0.419 -0.591

#> # ... with 4 more variables: vs , hp , carb , qsec

Or shave() the upper/lower triangle to missing values

rs %>% shave()

#> # A tibble: 11 x 12

#> rowname mpg cyl disp hp drat wt qsec

#>

#> 1 mpg NA NA NA NA NA NA NA

#> 2 cyl -0.852 NA NA NA NA NA NA

#> 3 disp -0.848 0.902 NA NA NA NA NA

#> 4 hp -0.775 0.830 0.786 NA NA NA NA

#> 5 drat 0.681 -0.700 -0.710 -0.443 NA NA NA

#> 6 wt -0.868 0.782 0.888 0.651 -0.712 NA NA

#> 7 qsec 0.419 -0.591 -0.434 -0.706 0.0912 -0.175 NA

#> 8 vs 0.664 -0.811 -0.710 -0.716 0.440 -0.555 0.745

#> 9 am 0.600 -0.523 -0.591 -0.220 0.713 -0.692 -0.230

#> 10 gear 0.480 -0.493 -0.556 -0.116 0.700 -0.583 -0.213

#> 11 carb -0.551 0.527 0.395 0.744 -0.0908 0.428 -0.656

#> # ... with 4 more variables: vs , am , gear , carb

Or stretch() into a more tidy format

rs %>% stretch()

#> # A tibble: 121 x 3

#> x y r

#>

#> 1 mpg mpg NA

#> 2 mpg cyl -0.852

#> 3 mpg disp -0.848

#> 4 mpg hp -0.775

#> 5 mpg drat 0.681

#> 6 mpg wt -0.868

#> 7 mpg qsec 0.419

#> 8 mpg vs 0.664

#> 9 mpg am 0.600

#> 10 mpg gear 0.480

#> # ... with 111 more rows

And combine with tidyverse to do things like get a histogram of all correlations:

rs %>%

shave() %>%

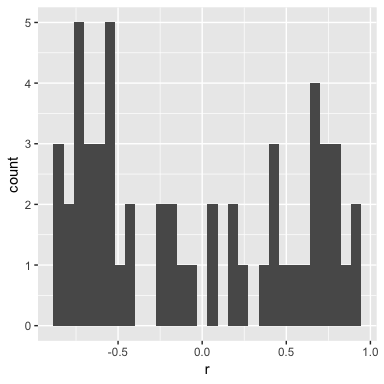
stretch(na.rm = FALSE) %>%

ggplot(aes(r)) +

geom\_histogram()

#> `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

#> Warning: Removed 66 rows containing non-finite values (stat\_bin).

[](https://i1.wp.com/svbtleusercontent.com/pA7uBfAFLB6dF3YTyL8RmF0xspap.png?ssl=1)

As a tidyverse-style package, it's important that the functions take a **data.frame in, data.frame out** principle. This let's you flow through pipelines and intermix functions from many packages with ease.

rs %>%

focus(mpg:drat, mirror = TRUE) %>%

rearrange() %>%

shave(upper = FALSE) %>%

select(-hp) %>%

filter(rowname != "drat")

#> # A tibble: 4 x 5

#> rowname cyl mpg disp drat

#>

#> 1 hp 0.830 -0.775 0.786 -0.443

#> 2 cyl NA -0.852 0.902 -0.700

#> 3 mpg NA NA -0.848 0.681

#> 4 disp NA NA NA -0.710

**Seems cool, but it's still hard to get quick insights**

corrr also provides some helpful methods to interpret/visualize the correlations. You can get fashionable:

rs %>% fashion()

#> rowname mpg cyl disp hp drat wt qsec vs am gear carb

#> 1 mpg -.85 -.85 -.77 .68 -.87 .42 .66 .60 .48 -.55

#> 2 cyl -.85 .90 .83 -.70 .78 -.59 -.81 -.52 -.49 .53

#> 3 disp -.85 .90 .79 -.71 .89 -.43 -.71 -.59 -.56 .39

#> 4 hp -.77 .83 .79 -.44 .65 -.71 -.72 -.22 -.12 .74

#> 5 drat .68 -.70 -.71 -.44 -.71 .09 .44 .71 .70 -.09

#> 6 wt -.87 .78 .89 .65 -.71 -.17 -.55 -.69 -.58 .43

#> 7 qsec .42 -.59 -.43 -.71 .09 -.17 .74 -.23 -.21 -.66

#> 8 vs .66 -.81 -.71 -.72 .44 -.55 .74 .17 .21 -.57

#> 9 am .60 -.52 -.59 -.22 .71 -.69 -.23 .17 .79 .06

#> 10 gear .48 -.49 -.56 -.12 .70 -.58 -.21 .21 .79 .27

#> 11 carb -.55 .53 .39 .74 -.09 .43 -.66 -.57 .06 .27

rs %>%

focus(mpg:drat, mirror = TRUE) %>%

rearrange() %>%

shave(upper = FALSE) %>%

select(-hp) %>%

filter(rowname != "drat") %>%

fashion()

#> rowname cyl mpg disp drat

#> 1 hp .83 -.77 .79 -.44

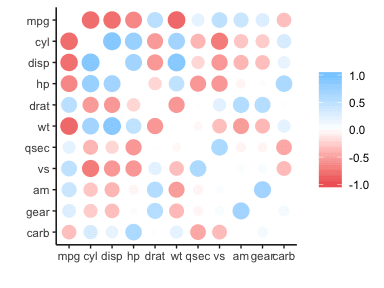
#> 2 cyl -.85 .90 -.70

#> 3 mpg -.85 .68

#> 4 disp -.71

Make an rplot()

rs %>% rplot()

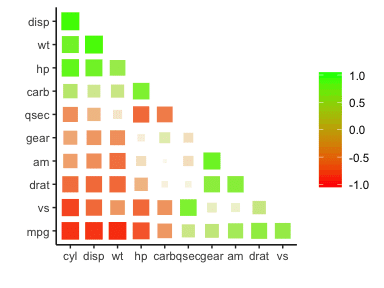
[](https://i0.wp.com/svbtleusercontent.com/vSUUWL5Ur8piZsWQRoqNAb0xspap.png?ssl=1)

rs %>%

rearrange(method = "MDS", absolute = FALSE) %>%

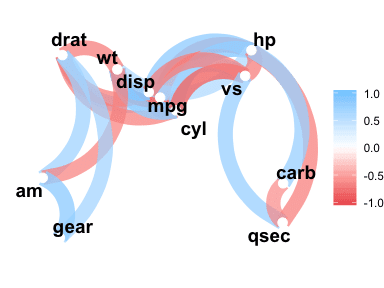
shave() %>%

rplot(shape = 15, colors = c("red", "green"))

[](https://i0.wp.com/svbtleusercontent.com/tWqQrSjZDakStTWqhuVNk60xspap.png?ssl=1)

Or make a network\_plot()

rs %>% network\_plot(min\_cor = .6)

[](https://i2.wp.com/svbtleusercontent.com/rhRX4PHzybDmmtXi2tfwYY0xspap.png?ssl=1)

But if you want to get custom, check out [my blog post combining corrr with ggraph](https://drsimonj.svbtle.com/how-to-create-correlation-network-plots-with-corrr-and-ggraph).

**Latest addition**

One of the latest editions by Edgar Ruiz lets you correlate() data frames in databases. To demonstrate (copying Edgar's vignette), here's a simple SQLite database with data pointer, db\_mtcars:

con <- DBI::dbConnect(RSQLite::SQLite(), path = ":dbname:")

db\_mtcars <- copy\_to(con, mtcars)

class(db\_mtcars)

#> [1] "tbl\_dbi" "tbl\_sql" "tbl\_lazy" "tbl"

correlate() detects DB backend, uses tidyeval to calculate correlations in the database, and returns correlation data frame.

db\_mtcars %>% correlate(use = "complete.obs")

#>

#> Correlation method: 'pearson'

#> Missing treated using: 'complete.obs'

#> # A tibble: 11 x 12

#> rowname mpg cyl disp hp drat wt qsec

#>

#> 1 mpg NA -0.852 -0.848 -0.776 0.681 -0.868 0.419

#> 2 cyl -0.852 NA 0.902 0.832 -0.700 0.782 -0.591

#> 3 disp -0.848 0.902 NA 0.791 -0.710 0.888 -0.434

#> 4 hp -0.776 0.832 0.791 NA -0.449 0.659 -0.708

#> 5 drat 0.681 -0.700 -0.710 -0.449 NA -0.712 0.0912

#> 6 wt -0.868 0.782 0.888 0.659 -0.712 NA -0.175

#> 7 qsec 0.419 -0.591 -0.434 -0.708 0.0912 -0.175 NA

#> 8 vs 0.664 -0.811 -0.710 -0.723 0.440 -0.555 0.745

#> 9 am 0.600 -0.523 -0.591 -0.243 0.713 -0.692 -0.230

#> 10 gear 0.480 -0.493 -0.556 -0.126 0.700 -0.583 -0.213

#> 11 carb -0.551 0.527 0.395 0.750 -0.0908 0.428 -0.656

#> # ... with 4 more variables: vs , am , gear , carb

Here's another example using spark:

sc <- sparklyr::spark\_connect(master = "local")

#> \* Using Spark: 2.1.0

mtcars\_tbl <- copy\_to(sc, mtcars)

correlate(mtcars\_tbl, use = "complete.obs")

#>

#> Correlation method: 'pearson'

#> Missing treated using: 'complete.obs'

#> # A tibble: 11 x 12

#> rowname mpg cyl disp hp drat wt qsec

#>

#> 1 mpg NA -0.852 -0.848 -0.776 0.681 -0.868 0.419

#> 2 cyl -0.852 NA 0.902 0.832 -0.700 0.782 -0.591

#> 3 disp -0.848 0.902 NA 0.791 -0.710 0.888 -0.434

#> 4 hp -0.776 0.832 0.791 NA -0.449 0.659 -0.708

#> 5 drat 0.681 -0.700 -0.710 -0.449 NA -0.712 0.0912

#> 6 wt -0.868 0.782 0.888 0.659 -0.712 NA -0.175

#> 7 qsec 0.419 -0.591 -0.434 -0.708 0.0912 -0.175 NA

#> 8 vs 0.664 -0.811 -0.710 -0.723 0.440 -0.555 0.745

#> 9 am 0.600 -0.523 -0.591 -0.243 0.713 -0.692 -0.230

#> 10 gear 0.480 -0.493 -0.556 -0.126 0.700 -0.583 -0.213

#> 11 carb -0.551 0.527 0.395 0.750 -0.0908 0.428 -0.656

#> # ... with 4 more variables: vs , am , gear , carb

So no data is too big for corrr now! This opens up some nice possibilities. For example, most regression-based modelling packages (like [lavaan](http://lavaan.ugent.be/)) cannot operate on large data sets in a database. However, they typically accept a correlation matrix as input. So you can use corrr to extract correlations from large data sets and do more complex modelling in memory.

In case corrr doesn't float your boat, some other packages you might be interested in are corrplot for rplot() style viz, and widyr for a more general way to handle relational data sets in a tidy framework.