431 Class 22

https://thomaselove.github.io/431-2024/

2024-11-14

## Today’s Agenda

* The here() package
* Turning values like 77 and 99 into NA and vice versa
* Selecting a multiple linear regression model for prediction of a quantitative outcome
* Incorporating both single and multiple imputation to handle missing data
* Our example: the msleep data from the ggplot2 package in the tidyverse

## Today’s Packages

library(here)  
library(naniar)  
library(janitor)  
library(broom)  
library(gt)  
library(mosaic) ## some nice tools for summaries  
library(car)  
library(GGally)  
library(mice)  
library(xfun)   
library(easystats)  
library(tidyverse)  
  
theme\_set(theme\_bw())

# The here() package

## The here() package

<https://here.r-lib.org/>

The **here** package creates paths relative to the top-level directory. The package displays the top-level of the current project on load or any time you call here():

here()

[1] "D:/Teaching/431/2024-431/431-slides-2024"

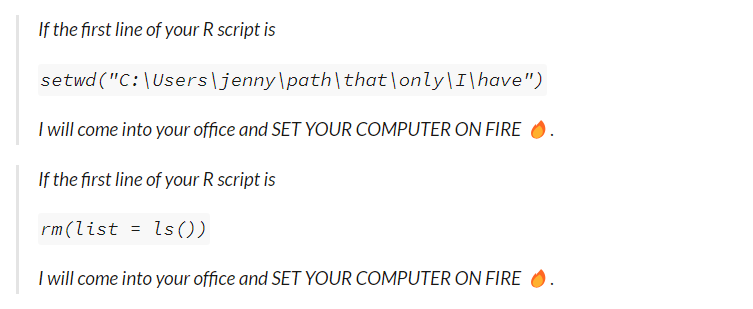
## The here() package



## Jenny Bryan’s [Ode to the here package](https://github.com/jennybc/here_here)



## Jenny Bryan on [Project-oriendted workflow](https://www.tidyverse.org/blog/2017/12/workflow-vs-script/)



## Using R projects and the here package

How can you avoid setwd() at the top of every script?

1. Organize each logical project into a folder on your computer.
2. Make sure the top-level folder advertises itself as such. If you use RStudio and/or Git, those both leave characteristic files that get the job done.
3. Use the here() function to build the path when you read or write a file. Create paths relative to the top-level directory.
4. Whenever you work on this project, launch the R process from the project’s top-level directory.

# Creating / Replacing Missing Values

## Turning values like 77 and 99 into NA

Suppose we have the following small data set, where 77 = “Refused” and 99 = “No Response” or some other term that we want to think of as “missing”.

var1 <- c(20, 22, 35, 19, 77, 99)  
var2 <- c(1, 3, 4, 77, 6, 99)  
var3 <- c("Yes", "No", 77, 99, "No", "Yes")  
dat <- tibble(var1, var2, var3) |> mutate(var3 = factor(var3))  
  
miss\_var\_summary(dat)

# A tibble: 3 × 3  
 variable n\_miss pct\_miss  
 <chr> <int> <num>  
1 var1 0 0  
2 var2 0 0  
3 var3 0 0

How can we convince R the 77s and 99s are missing values?

## Use replace\_with\_na() from naniar

dat1 <- dat |>  
 replace\_with\_na(  
 replace = list(var1 = c(77, 99), var2 = c(77, 99),  
 var3 = c(77, 99)))  
miss\_var\_summary(dat1)

# A tibble: 3 × 3  
 variable n\_miss pct\_miss  
 <chr> <int> <num>  
1 var1 2 33.3  
2 var2 2 33.3  
3 var3 2 33.3

More on replace\_with\_na() [here](https://cran.r-project.org/web/packages/naniar/vignettes/replace-with-na.html)

## Replacing 77 and 99 with NA across all variables

dat2 <- dat |>  
 replace\_with\_na\_all(  
 condition = ~.x %in% c(77, 99))  
miss\_var\_summary(dat2)

# A tibble: 3 × 3  
 variable n\_miss pct\_miss  
 <chr> <int> <num>  
1 var1 2 33.3  
2 var2 2 33.3  
3 var3 2 33.3

Other ways to extend replace\_with\_na() are described [here](https://cran.r-project.org/web/packages/naniar/vignettes/replace-with-na.html)

## What if we have the opposite issue?

The replace\_na() function from the tidyr package ([details here](https://tidyr.tidyverse.org/reference/replace_na.html)) replaces an NA value with a specified value.

In that sense, it is the compliment to the replace\_with\_na() function.

## Demo: Replacing NA with a value

df <- tibble(x = c(1, 2, NA), y = c("a", NA, "b"))  
df

# A tibble: 3 × 2  
 x y   
 <dbl> <chr>  
1 1 a   
2 2 <NA>   
3 NA b

df1 <- df |> replace\_na(list(x = 0, y = "unknown"))  
df1

# A tibble: 3 × 2  
 x y   
 <dbl> <chr>   
1 1 a   
2 2 unknown  
3 0 b

More on replace\_na() [here](https://tidyr.tidyverse.org/reference/replace_na.html)

# Mammals and How They Sleep

## The mammals sleep data set (msleep)

msleep

# A tibble: 83 × 11  
 name genus vore order conservation sleep\_total sleep\_rem sleep\_cycle awake  
 <chr> <chr> <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl>  
 1 Cheet… Acin… carni Carn… lc 12.1 NA NA 11.9  
 2 Owl m… Aotus omni Prim… <NA> 17 1.8 NA 7   
 3 Mount… Aplo… herbi Rode… nt 14.4 2.4 NA 9.6  
 4 Great… Blar… omni Sori… lc 14.9 2.3 0.133 9.1  
 5 Cow Bos herbi Arti… domesticated 4 0.7 0.667 20   
 6 Three… Brad… herbi Pilo… <NA> 14.4 2.2 0.767 9.6  
 7 North… Call… carni Carn… vu 8.7 1.4 0.383 15.3  
 8 Vespe… Calo… <NA> Rode… <NA> 7 NA NA 17   
 9 Dog Canis carni Carn… domesticated 10.1 2.9 0.333 13.9  
10 Roe d… Capr… herbi Arti… lc 3 NA NA 21   
# ℹ 73 more rows  
# ℹ 2 more variables: brainwt <dbl>, bodywt <dbl>

* <https://ggplot2.tidyverse.org/reference/msleep.html> has details.

## Today’s Variables of Interest

| Variable | Description |
| --- | --- |
| name | Common name of mammal |
| vore | Carni-, insecti-, herbi- or omnivore |
| brainwt | brain weight in kilograms |
| bodywt | body weight in kilograms |
| sleep\_rem | REM sleep, in hours |
| sleep\_total | total amount of sleep, in hours |
| awake | (**outcome**) time spent awake, in hours |

## Modeling Plan

|  |
| --- |
| Note |
| Note that awake = 24 - sleep\_total. |

We want to predict awake using four potential predictors:

* the type of food the mammal consumes (vore)
* the weight of the mammal’s brain (brainwt)
* the weight of the mammal’s body, (bodywt), and
* the proportion of total time asleep spent in REM sleep.

## Creating the ms431 data

ms431 <- msleep |>  
 mutate(rem\_prop = sleep\_rem / sleep\_total) |>  
 mutate(vore = factor(vore)) |>  
 select(name, vore, brainwt, bodywt, rem\_prop, awake)  
  
glimpse(ms431)

Rows: 83  
Columns: 6  
$ name <chr> "Cheetah", "Owl monkey", "Mountain beaver", "Greater short-ta…  
$ vore <fct> carni, omni, herbi, omni, herbi, herbi, carni, NA, carni, her…  
$ brainwt <dbl> NA, 0.01550, NA, 0.00029, 0.42300, NA, NA, NA, 0.07000, 0.098…  
$ bodywt <dbl> 50.000, 0.480, 1.350, 0.019, 600.000, 3.850, 20.490, 0.045, 1…  
$ rem\_prop <dbl> NA, 0.10588235, 0.16666667, 0.15436242, 0.17500000, 0.1527777…  
$ awake <dbl> 11.9, 7.0, 9.6, 9.1, 20.0, 9.6, 15.3, 17.0, 13.9, 21.0, 18.7,…

## Exploring our factor variable

ms431 |> tabyl(vore) |> adorn\_pct\_formatting() |> gt()

| vore | n | percent | valid\_percent |
| --- | --- | --- | --- |
| carni | 19 | 22.9% | 25.0% |
| herbi | 32 | 38.6% | 42.1% |
| insecti | 5 | 6.0% | 6.6% |
| omni | 20 | 24.1% | 26.3% |
| NA | 7 | 8.4% | - |

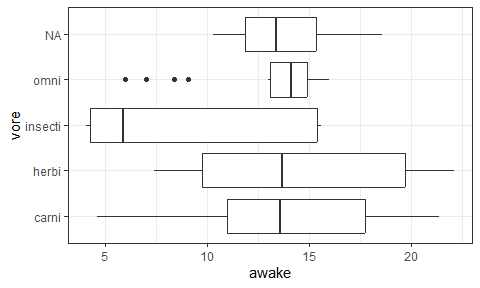
Any concerns here?

## Collapse to three vore groups?

favstats(awake ~ vore, data = ms431) |>  
 gt() |> fmt\_number(columns = min:sd, decimals = 3)

| vore | min | Q1 | median | Q3 | max | mean | sd | n | missing |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| carni | 4.600 | 11.000 | 13.600 | 17.750 | 21.350 | 13.626 | 4.677 | 19 | 0 |
| herbi | 7.400 | 9.775 | 13.700 | 19.700 | 22.100 | 14.491 | 4.879 | 32 | 0 |
| insecti | 4.100 | 4.300 | 5.900 | 15.400 | 15.600 | 9.060 | 5.921 | 5 | 0 |
| omni | 6.000 | 13.075 | 14.100 | 14.900 | 16.000 | 13.075 | 2.949 | 20 | 0 |

ggplot(ms431, aes(x = awake, y = vore)) + geom\_boxplot()



## Exploring our quantities

df\_stats(~ brainwt + bodywt + rem\_prop + awake, data = ms431) |>  
 gt() |> fmt\_number(columns = min:sd, decimals = 3)

| response | min | Q1 | median | Q3 | max | mean | sd | n | missing |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| brainwt | 0.000 | 0.003 | 0.012 | 0.126 | 5.712 | 0.282 | 0.976 | 56 | 27 |
| bodywt | 0.005 | 0.174 | 1.670 | 41.750 | 6,654.000 | 166.136 | 786.840 | 83 | 0 |
| rem\_prop | 0.037 | 0.113 | 0.156 | 0.227 | 0.347 | 0.174 | 0.072 | 61 | 22 |
| awake | 4.100 | 10.250 | 13.900 | 16.150 | 22.100 | 13.567 | 4.452 | 83 | 0 |

Any concerns here?

## A brainwt of 0?

which.min(ms431$brainwt)

[1] 17

slice(ms431, 17)

# A tibble: 1 × 6  
 name vore brainwt bodywt rem\_prop awake  
 <chr> <fct> <dbl> <dbl> <dbl> <dbl>  
1 Lesser short-tailed shrew omni 0.00014 0.005 0.154 14.9

* How many of these mammals have a brainwt below 0.01 kg?

ms431 |> filter(brainwt < 0.01) |> nrow()

[1] 23

## Collinearity Check

vif(lm(awake ~ vore + bodywt + brainwt + rem\_prop, data = ms431))

GVIF Df GVIF^(1/(2\*Df))  
vore 1.571683 3 1.078270  
bodywt 1.658313 1 1.287755  
brainwt 1.459939 1 1.208280  
rem\_prop 1.310584 1 1.144807

cor(ms431$bodywt, ms431$brainwt, use = "complete.obs")

[1] 0.9337822

* What can we do about this?

## Change our set of variables?

What if we included

* brain\_prop: brain weight as a proportion of body weight

along with bodywt in our model?

ms431 <- ms431 |>  
 mutate(brain\_prop = brainwt / bodywt) |>  
 select(name, vore, bodywt, brain\_prop, rem\_prop, awake)  
  
vif(lm(awake ~ vore + bodywt + brain\_prop + rem\_prop, data = ms431))

GVIF Df GVIF^(1/(2\*Df))  
vore 1.685346 3 1.090891  
bodywt 1.277174 1 1.130121  
brain\_prop 1.330301 1 1.153387  
rem\_prop 1.346185 1 1.160252

OK. Let’s move on to think about missingness.

## How much missingness do we have?

miss\_var\_summary(ms431)

# A tibble: 6 × 3  
 variable n\_miss pct\_miss  
 <chr> <int> <num>  
1 brain\_prop 27 32.5   
2 rem\_prop 22 26.5   
3 vore 7 8.43  
4 name 0 0   
5 bodywt 0 0   
6 awake 0 0

miss\_case\_table(ms431)

# A tibble: 4 × 3  
 n\_miss\_in\_case n\_cases pct\_cases  
 <int> <int> <dbl>  
1 0 43 51.8   
2 1 26 31.3   
3 2 12 14.5   
4 3 2 2.41

## Missing Data Mechanisms

* **Missing completely at random** There are no systematic differences between the missing values and the observed values.
  + For example, blood pressure measurements may be missing because of breakdown of an automatic sphygmomanometer.
* **Missing at random** Any systematic difference between the missing and observed values can be explained by other observed data.
  + For example, missing BP measurements may be lower than measured BPs but only because younger people more often have a missing BP.
* **Missing not at random** Even after the observed data are taken into account, systematic differences remain between the missing values and the observed values.
  + For example, people with high BP may be more likely to have headaches that cause them to miss clinic appointments.

“Missing at random” is an **assumption** that justifies the analysis, rather than a property of the data.

## What assumption should we use?

Can we assume the data are MCAR, per Little’s test?

mcar\_test(ms431) ## naniar provides this function

# A tibble: 1 × 4  
 statistic df p.value missing.patterns  
 <dbl> <dbl> <dbl> <int>  
1 33.5 22 0.0549 6

With a small value for the test statistic, we would conclude that the ms431 data are not MCAR.

|  |
| --- |
| Reference |
| Little, Roderick J. A. 1988. “A Test of Missing Completely at Random for Multivariate Data with Missing Values.” *Journal of the American Statistical Association* 83 (404): 1198–1202. [doi:10.1080/01621459.1988.10478722](https://www.tandfonline.com/doi/abs/10.1080/01621459.1988.10478722). |

## If not MCAR, then what?

Suppose we assume that the data are MAR. This suggests the need for imputation of missing values.

|  |
| --- |
| Note |
| If we were willing to assume MCAR, we could simply do a complete case analysis. |

Here, we have complete data on our outcome (awake) and bodywt so we won’t need to impute them.

* We will need to impute vore (7), brain\_prop (27) and rem\_prop (22) from our sample of 83 mammals.
* vore is a factor, the others are quantities.

## Single Imputation

We’ll create a singly imputed data set first, to select our predictors.

set.seed(12345)  
ms431\_imp1 <- mice(ms431, m = 1, printFlag = FALSE)

Warning: Number of logged events: 9

ms431\_imp <- complete(ms431\_imp1)  
  
prop\_complete(ms431); prop\_complete(ms431\_imp)

[1] 0.8875502

[1] 1

|  |
| --- |
| Note |
| After we’ve settled on a final prediction model using ms431\_imp, we’ll implement multiple imputation. |

## Range Check for impossible values?

ms431\_imp |> tabyl(vore) |> adorn\_pct\_formatting() |> gt()

| vore | n | percent |
| --- | --- | --- |
| carni | 19 | 22.9% |
| herbi | 36 | 43.4% |
| insecti | 5 | 6.0% |
| omni | 23 | 27.7% |

df\_stats(~ brain\_prop + bodywt + rem\_prop + awake, data = ms431\_imp) |>  
 gt() |> fmt\_number(columns = min:sd, decimals = 3)

| response | min | Q1 | median | Q3 | max | mean | sd | n | missing |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| brain\_prop | 0.001 | 0.003 | 0.008 | 0.017 | 0.040 | 0.011 | 0.009 | 83 | 0 |
| bodywt | 0.005 | 0.174 | 1.670 | 41.750 | 6,654.000 | 166.136 | 786.840 | 83 | 0 |
| rem\_prop | 0.037 | 0.113 | 0.158 | 0.237 | 0.347 | 0.175 | 0.071 | 83 | 0 |
| awake | 4.100 | 10.250 | 13.900 | 16.150 | 22.100 | 13.567 | 4.452 | 83 | 0 |

## Partitioning the ms431\_imp data

* Do we have a unique name to identify each mammal?

n\_distinct(ms431\_imp$name) == nrow(ms431\_imp) # compare numbers

[1] TRUE

near(n\_distinct(ms431\_imp$name), nrow(ms431\_imp)) # tidyverse approach

[1] TRUE

* Partition 70% into training sample, remaining 30% to test, while maintaining a similar percentage by vore groups.

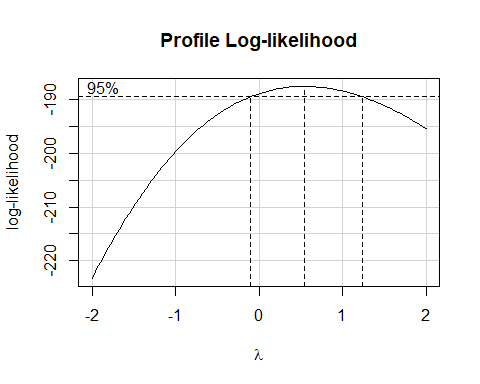
set.seed(20231128)  
ms431\_train <- slice\_sample(ms431\_imp, prop = 0.7, by = "vore")  
ms431\_test <- anti\_join(ms431\_imp, ms431\_train, by = "name")  
  
dim(ms431\_train); dim(ms431\_test)

[1] 57 6

[1] 26 6

## Outcome transformation?

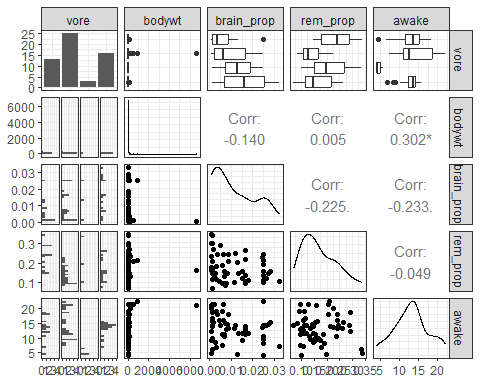
boxCox(lm(awake ~ vore + bodywt + brain\_prop + rem\_prop, data = ms431\_train))



## Scatterplot Matrix

ggpairs(ms431\_train |> select(vore, bodywt, brain\_prop, rem\_prop, awake))

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



## Collinearity Check in Training Sample

vif(lm(awake ~ vore + bodywt + brain\_prop + rem\_prop, data = ms431\_train))

GVIF Df GVIF^(1/(2\*Df))  
vore 1.725015 3 1.095129  
bodywt 1.056919 1 1.028066  
brain\_prop 1.207572 1 1.098896  
rem\_prop 1.531482 1 1.237531

## Which Potential Models Will We Fit?

* Model 1: A simple regression on bodywt
* Model 2: A simple regression on brain\_prop
* Model 3: A model with the two size variables (brain\_prop and bodywt)
* Model 4: Model 3 + vore
* Model 5: Model 3 + rem\_prop
* Model 6: All four predictors

## Model 6

m6 <- lm(awake ~ bodywt + brain\_prop + vore + rem\_prop, data = ms431\_train)  
model\_parameters(m6, ci = 0.90) |> gt()

| Parameter | Coefficient | SE | CI | CI\_low | CI\_high | t | df\_error | p |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 12.861509015 | 2.537865e+00 | 0.9 | 8.608288e+00 | 17.114730227 | 5.0678454 | 50 | 5.878093e-06 |
| bodywt | 0.001204781 | 5.970149e-04 | 0.9 | 2.042404e-04 | 0.002205321 | 2.0180077 | 50 | 4.897220e-02 |
| brain\_prop | -83.198697107 | 6.012258e+01 | 0.9 | -1.839584e+02 | 17.561029934 | -1.3838179 | 50 | 1.725609e-01 |
| voreherbi | 1.442092605 | 1.601502e+00 | 0.9 | -1.241872e+00 | 4.126057298 | 0.9004628 | 50 | 3.721908e-01 |
| voreinsecti | -7.614590737 | 2.500228e+00 | 0.9 | -1.180474e+01 | -3.424445527 | -3.0455581 | 50 | 3.700684e-03 |
| voreomni | 0.753316104 | 1.637686e+00 | 0.9 | -1.991290e+00 | 3.497922240 | 0.4599881 | 50 | 6.475188e-01 |
| rem\_prop | 2.776453501 | 9.496943e+00 | 0.9 | -1.313952e+01 | 18.692428736 | 0.2923523 | 50 | 7.712270e-01 |

model\_performance(m6) |> gt()

| AIC | AICc | BIC | R2 | R2\_adjusted | RMSE | Sigma |
| --- | --- | --- | --- | --- | --- | --- |
| 324.0256 | 327.0256 | 340.37 | 0.322219 | 0.2408853 | 3.607592 | 3.851854 |

## Model 5

m5 <- lm(awake ~ bodywt + brain\_prop + rem\_prop, data = ms431\_train)  
model\_parameters(m5, ci = 0.90) |> gt()

| Parameter | Coefficient | SE | CI | CI\_low | CI\_high | t | df\_error | p |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 1.507494e+01 | 1.850906e+00 | 0.9 | 1.197631e+01 | 18.17357317 | 8.1446283 | 53 | 6.578536e-11 |
| bodywt | 1.359156e-03 | 6.426695e-04 | 0.9 | 2.832530e-04 | 0.00243506 | 2.1148606 | 53 | 3.916124e-02 |
| brain\_prop | -1.022174e+02 | 6.214353e+01 | 0.9 | -2.062529e+02 | 1.81807281 | -1.6448602 | 53 | 1.059177e-01 |
| rem\_prop | -6.547216e+00 | 8.630335e+00 | 0.9 | -2.099540e+01 | 7.90096864 | -0.7586282 | 53 | 4.514356e-01 |

model\_performance(m5) |> gt()

| AIC | AICc | BIC | R2 | R2\_adjusted | RMSE | Sigma |
| --- | --- | --- | --- | --- | --- | --- |
| 331.7264 | 332.9029 | 341.9417 | 0.1380569 | 0.08926767 | 4.068291 | 4.219019 |

## Model 4

m4 <- lm(awake ~ bodywt + brain\_prop + vore, data = ms431\_train)  
model\_parameters(m4, ci = 0.90) |> gt()

| Parameter | Coefficient | SE | CI | CI\_low | CI\_high | t | df\_error | p |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 13.524618518 | 1.128207e+00 | 0.9 | 1.163455e+01 | 15.414686173 | 11.9877137 | 51 | 1.875095e-16 |
| bodywt | 0.001217627 | 5.900331e-04 | 0.9 | 2.291540e-04 | 0.002206101 | 2.0636597 | 51 | 4.415652e-02 |
| brain\_prop | -86.220983537 | 5.869370e+01 | 0.9 | -1.845497e+02 | 12.107694999 | -1.4689989 | 51 | 1.479748e-01 |
| voreherbi | 1.188933544 | 1.335070e+00 | 0.9 | -1.047690e+00 | 3.425556767 | 0.8905400 | 51 | 3.773579e-01 |
| voreinsecti | -7.650136315 | 2.474779e+00 | 0.9 | -1.179610e+01 | -3.504177125 | -3.0912408 | 51 | 3.226971e-03 |
| voreomni | 0.587367948 | 1.522332e+00 | 0.9 | -1.962972e+00 | 3.137707806 | 0.3858343 | 51 | 7.012242e-01 |

model\_performance(m4) |> gt()

| AIC | AICc | BIC | R2 | R2\_adjusted | RMSE | Sigma |
| --- | --- | --- | --- | --- | --- | --- |
| 322.123 | 324.4087 | 336.4243 | 0.3210604 | 0.2544977 | 3.610674 | 3.817162 |

## Model 3

m3 <- lm(awake ~ bodywt + brain\_prop, data = ms431\_train)  
model\_parameters(m3, ci = 0.90) |> gt()

| Parameter | Coefficient | SE | CI | CI\_low | CI\_high | t | df\_error | p |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 13.841315609 | 8.805796e-01 | 0.9 | 1.236761e+01 | 15.315022680 | 15.718415 | 54 | 8.426394e-22 |
| bodywt | 0.001372401 | 6.399023e-04 | 0.9 | 3.014830e-04 | 0.002443319 | 2.144704 | 54 | 3.649326e-02 |
| brain\_prop | -91.532194409 | 6.028794e+01 | 0.9 | -1.924280e+02 | 9.363581154 | -1.518251 | 54 | 1.347846e-01 |

model\_performance(m3) |> gt()

| AIC | AICc | BIC | R2 | R2\_adjusted | RMSE | Sigma |
| --- | --- | --- | --- | --- | --- | --- |
| 330.342 | 331.1113 | 338.5142 | 0.1286972 | 0.09642676 | 4.09032 | 4.202404 |

## Model 2

m2 <- lm(awake ~ brain\_prop, data = ms431\_train)  
model\_parameters(m2, ci = 0.90) |> gt()

| Parameter | Coefficient | SE | CI | CI\_low | CI\_high | t | df\_error | p |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 14.26293 | 0.8860011 | 0.9 | 12.78062 | 15.745236 | 16.098091 | 55 | 1.814412e-22 |
| brain\_prop | -109.68223 | 61.6134523 | 0.9 | -212.76363 | -6.600831 | -1.780167 | 55 | 8.057277e-02 |

model\_performance(m2) |> gt()

| AIC | AICc | BIC | R2 | R2\_adjusted | RMSE | Sigma |
| --- | --- | --- | --- | --- | --- | --- |
| 333.0016 | 333.4544 | 339.1307 | 0.0544791 | 0.03728781 | 4.260968 | 4.337748 |

## Model 1

m1 <- lm(awake ~ bodywt, data = ms431\_train)  
model\_parameters(m1, ci = 0.90) |> gt()

| Parameter | Coefficient | SE | CI | CI\_low | CI\_high | t | df\_error | p |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 12.817198699 | 0.5727327039 | 0.9 | 1.18590e+01 | 13.775399966 | 22.379024 | 55 | 2.665922e-29 |
| bodywt | 0.001508777 | 0.0006410394 | 0.9 | 4.36296e-04 | 0.002581257 | 2.353641 | 55 | 2.219168e-02 |

model\_performance(m1) |> gt()

| AIC | AICc | BIC | R2 | R2\_adjusted | RMSE | Sigma |
| --- | --- | --- | --- | --- | --- | --- |
| 330.7247 | 331.1775 | 336.8538 | 0.09150414 | 0.07498604 | 4.176709 | 4.251971 |

## Combining glance() results

bind\_rows(glance(m1) |> mutate(name = "m1"),  
 glance(m2) |> mutate(name = "m2"),  
 glance(m3) |> mutate(name = "m3"),  
 glance(m4) |> mutate(name = "m4"),  
 glance(m5) |> mutate(name = "m5"),  
 glance(m6) |> mutate(name = "m6")) |>  
 select(name, r.squared, adj.r.squared, sigma, AIC, BIC, nobs, df) |>  
 gt() |> fmt\_number(decimals = 5, columns = 2) |>  
 fmt\_number(decimals = 4, columns = c(3:4)) |>  
 fmt\_number(decimals = 0, columns = c(5:6))

| name | r.squared | adj.r.squared | sigma | AIC | BIC | nobs | df |
| --- | --- | --- | --- | --- | --- | --- | --- |
| m1 | 0.09150 | 0.0750 | 4.2520 | 331 | 337 | 57 | 1 |
| m2 | 0.05448 | 0.0373 | 4.3377 | 333 | 339 | 57 | 1 |
| m3 | 0.12870 | 0.0964 | 4.2024 | 330 | 339 | 57 | 2 |
| m4 | 0.32106 | 0.2545 | 3.8172 | 322 | 336 | 57 | 5 |
| m5 | 0.13806 | 0.0893 | 4.2190 | 332 | 342 | 57 | 3 |
| m6 | 0.32222 | 0.2409 | 3.8519 | 324 | 340 | 57 | 6 |

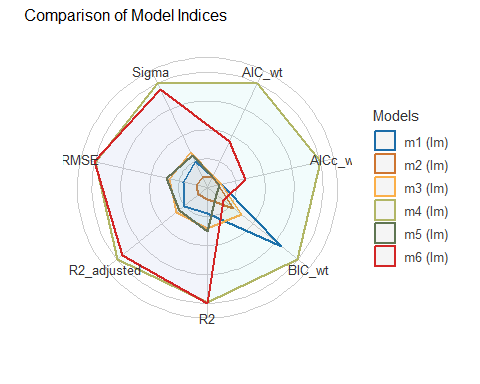
## Compare Models

compare\_performance(m1, m2, m3, m4, m5, m6, rank = TRUE)

# Comparison of Model Performance Indices  
  
Name | Model | R2 | R2 (adj.) | RMSE | Sigma | AIC weights | AICc weights | BIC weights | Performance-Score  
---------------------------------------------------------------------------------------------------------------  
m4 | lm | 0.321 | 0.254 | 3.611 | 3.817 | 0.700 | 0.733 | 0.382 | 99.87%  
m6 | lm | 0.322 | 0.241 | 3.608 | 3.852 | 0.270 | 0.198 | 0.053 | 65.68%  
m1 | lm | 0.092 | 0.075 | 4.177 | 4.252 | 0.009 | 0.025 | 0.308 | 20.45%  
m3 | lm | 0.129 | 0.096 | 4.090 | 4.202 | 0.011 | 0.026 | 0.134 | 20.21%  
m5 | lm | 0.138 | 0.089 | 4.068 | 4.219 | 0.006 | 0.010 | 0.024 | 15.45%  
m2 | lm | 0.054 | 0.037 | 4.261 | 4.338 | 0.003 | 0.008 | 0.099 | 2.97%

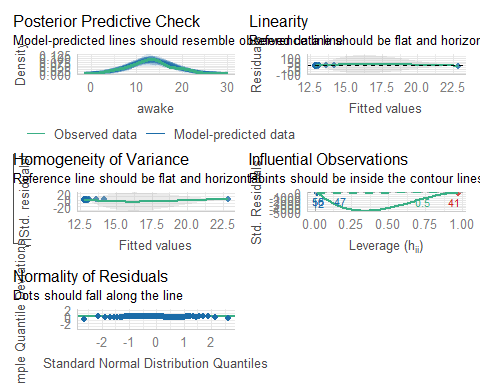
## Compare Models

plot(compare\_performance(m1, m2, m3, m4, m5, m6))



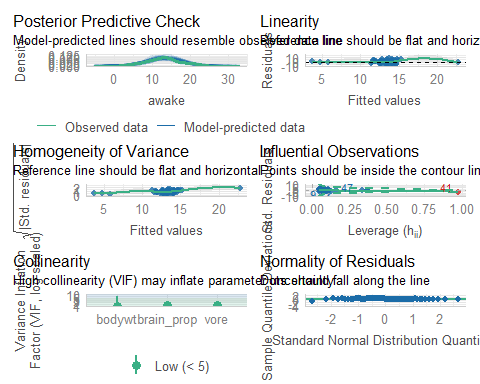
## Check Assumptions? (m1)

check\_model(m1)



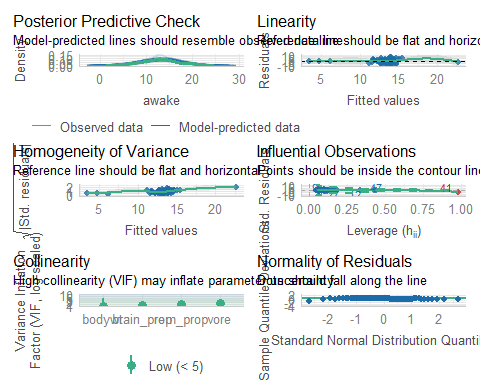
## Check Assumptions? (m4)

check\_model(m4)



## Check Assumptions? (m6)

check\_model(m6)



## Move forward with m1, m4 and m6

test\_m1 <- augment(m1, newdata = ms431\_test) |> mutate(mod = "m1")  
test\_m4 <- augment(m4, newdata = ms431\_test) |> mutate(mod = "m4")  
test\_m6 <- augment(m6, newdata = ms431\_test) |> mutate(mod = "m6")  
  
test\_comp <- bind\_rows(test\_m1, test\_m4, test\_m6) |>  
 arrange(name, mod)

## Comparing Models: Test Sample

test\_comp |>  
 group\_by(mod) |>  
 summarize(n = n(),  
 MAPE = mean(abs(.resid)),   
 RMSPE = sqrt(mean(.resid^2)),  
 max\_error = max(abs(.resid)),  
 valid\_R2 = cor(awake, .fitted)^2) |>  
 gt() |> fmt\_number(decimals = 4, columns = -"n")

| mod | n | MAPE | RMSPE | max\_error | valid\_R2 |
| --- | --- | --- | --- | --- | --- |
| m1 | 26 | 3.8788 | 4.3714 | 7.5530 | 0.1599 |
| m4 | 26 | 4.1504 | 5.0685 | 11.1050 | 0.0163 |
| m6 | 26 | 4.2407 | 5.1647 | 10.9901 | 0.0068 |

I’d probably pick model m1 or m4. But first I’ll show what happens if we pick model m6 instead, just because I want to show as much missingness as possible.

## Model 6 (training sample)

m6 <- lm(awake ~ bodywt + brain\_prop + vore + rem\_prop, data = ms431\_train)  
model\_parameters(m6, ci = 0.90) |> gt()

| Parameter | Coefficient | SE | CI | CI\_low | CI\_high | t | df\_error | p |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 12.861509015 | 2.537865e+00 | 0.9 | 8.608288e+00 | 17.114730227 | 5.0678454 | 50 | 5.878093e-06 |
| bodywt | 0.001204781 | 5.970149e-04 | 0.9 | 2.042404e-04 | 0.002205321 | 2.0180077 | 50 | 4.897220e-02 |
| brain\_prop | -83.198697107 | 6.012258e+01 | 0.9 | -1.839584e+02 | 17.561029934 | -1.3838179 | 50 | 1.725609e-01 |
| voreherbi | 1.442092605 | 1.601502e+00 | 0.9 | -1.241872e+00 | 4.126057298 | 0.9004628 | 50 | 3.721908e-01 |
| voreinsecti | -7.614590737 | 2.500228e+00 | 0.9 | -1.180474e+01 | -3.424445527 | -3.0455581 | 50 | 3.700684e-03 |
| voreomni | 0.753316104 | 1.637686e+00 | 0.9 | -1.991290e+00 | 3.497922240 | 0.4599881 | 50 | 6.475188e-01 |
| rem\_prop | 2.776453501 | 9.496943e+00 | 0.9 | -1.313952e+01 | 18.692428736 | 0.2923523 | 50 | 7.712270e-01 |

model\_performance(m6) |> gt()

| AIC | AICc | BIC | R2 | R2\_adjusted | RMSE | Sigma |
| --- | --- | --- | --- | --- | --- | --- |
| 324.0256 | 327.0256 | 340.37 | 0.322219 | 0.2408853 | 3.607592 | 3.851854 |

## Create Multiple Imputations

How many subjects have missing data that affect this model?

ms431\_sub <- ms431 |> select(name, awake, bodywt, brain\_prop, rem\_prop, vore)  
  
pct\_miss\_case(ms431\_sub)

[1] 48.19277

## We’ll build 50 imputed data sets.

set.seed(4312345)  
ms431\_mice <- mice(ms431, m = 50, printFlag = FALSE)

Warning: Number of logged events: 425

summary(ms431\_mice)

Class: mids  
Number of multiple imputations: 50   
Imputation methods:  
 name vore bodywt brain\_prop rem\_prop awake   
 "" "polyreg" "" "pmm" "pmm" ""   
PredictorMatrix:  
 name vore bodywt brain\_prop rem\_prop awake  
name 0 1 1 1 1 1  
vore 0 0 1 1 1 1  
bodywt 0 1 0 1 1 1  
brain\_prop 0 1 1 0 1 1  
rem\_prop 0 1 1 1 0 1  
awake 0 1 1 1 1 0  
Number of logged events: 425   
 it im dep meth out  
1 0 0 constant name  
2 1 1 rem\_prop pmm brain\_prop  
3 1 2 vore polyreg brain\_prop  
4 1 2 rem\_prop pmm brain\_prop  
5 1 3 vore polyreg brain\_prop  
6 1 3 rem\_prop pmm brain\_prop

## Run Model 6 on each imputation

m6\_mods <- with(ms431\_mice, lm(awake ~ bodywt + brain\_prop + vore + rem\_prop))  
  
summary(m6\_mods)

# A tibble: 350 × 6  
 term estimate std.error statistic p.value nobs  
 <chr> <dbl> <dbl> <dbl> <dbl> <int>  
 1 (Intercept) 16.2 1.80 9.00 1.35e-13 83  
 2 bodywt 0.00178 0.000622 2.86 5.50e- 3 83  
 3 brain\_prop -92.9 56.5 -1.64 1.04e- 1 83  
 4 voreherbi -0.104 1.26 -0.0824 9.35e- 1 83  
 5 voreinsecti -3.59 2.11 -1.70 9.32e- 2 83  
 6 voreomni 0.281 1.33 0.211 8.34e- 1 83  
 7 rem\_prop -10.7 7.29 -1.47 1.46e- 1 83  
 8 (Intercept) 16.7 1.73 9.66 7.25e-15 83  
 9 bodywt 0.00128 0.000588 2.18 3.26e- 2 83  
10 brain\_prop -124. 50.6 -2.45 1.65e- 2 83  
# ℹ 340 more rows

## Pool Results across imputations

m6\_pool <- pool(m6\_mods)  
model\_parameters(m6\_pool, ci = 0.90)

# Fixed Effects  
  
Parameter | Coefficient | SE | 90% CI | t | df | p  
-----------------------------------------------------------------------------------  
(Intercept) | 16.55 | 2.07 | [ 13.08, 20.02] | 7.99 | 49.67 | < .001  
bodywt | 1.57e-03 | 6.28e-04 | [ 0.00, 0.00] | 2.51 | 67.14 | 0.015   
brain prop | -86.43 | 62.91 | [-191.78, 18.92] | -1.37 | 51.91 | 0.175   
vore [herbi] | -0.20 | 1.29 | [ -2.35, 1.95] | -0.15 | 64.44 | 0.877   
vore [insecti] | -3.98 | 2.07 | [ -7.43, -0.54] | -1.93 | 69.69 | 0.058   
vore [omni] | -0.19 | 1.39 | [ -2.50, 2.12] | -0.14 | 64.60 | 0.891   
rem prop | -10.94 | 7.91 | [ -24.20, 2.31] | -1.38 | 48.85 | 0.173

Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed  
 using a Wald t-distribution approximation.

## Estimate R-square and Adjusted R-square

pool.r.squared(m6\_mods)

est lo 95 hi 95 fmi  
R^2 0.216375 0.07007382 0.3932251 0.1137024

pool.r.squared(m6\_mods, adjusted = TRUE)

est lo 95 hi 95 fmi  
adj R^2 0.154012 0.03068205 0.3286404 0.1508684

## More Details on MI modeling

m6\_pool

Class: mipo m = 50   
 term m estimate ubar b t dfcom  
1 (Intercept) 50 16.547867200 3.112943e+00 1.152888e+00 4.288889e+00 76  
2 bodywt 50 0.001574986 3.608544e-07 3.271760e-08 3.942264e-07 76  
3 brain\_prop 50 -86.427263339 2.968903e+03 9.687751e+02 3.957054e+03 76  
4 voreherbi 50 -0.199673328 1.469290e+00 1.866263e-01 1.659649e+00 76  
5 voreinsecti 50 -3.983083996 4.038323e+00 2.308205e-01 4.273759e+00 76  
6 voreomni 50 -0.190005602 1.703208e+00 2.126782e-01 1.920140e+00 76  
7 rem\_prop 50 -10.940624618 4.479904e+01 1.735075e+01 6.249680e+01 76  
 df riv lambda fmi  
1 49.66843 0.37776012 0.27418425 0.30174595  
2 67.13953 0.09248037 0.08465174 0.11075252  
3 51.90642 0.33283356 0.24971877 0.27704823  
4 64.44489 0.12955833 0.11469822 0.14095082  
5 69.69306 0.05830066 0.05508894 0.08108623  
6 64.60136 0.12736657 0.11297706 0.13921982  
7 48.85388 0.39504789 0.28317873 0.31082647

## What if we picked model m1?

m1\_mods <- with(ms431\_mice, lm(awake ~ bodywt))  
m1\_pool <- pool(m1\_mods)  
model\_parameters(m1\_pool, ci = 0.90)

# Fixed Effects  
  
Parameter | Coefficient | SE | 90% CI | t(79.06) | p  
-------------------------------------------------------------------------  
(Intercept) | 13.27 | 0.48 | [12.48, 14.07] | 27.80 | < .001  
bodywt | 1.77e-03 | 5.97e-04 | [ 0.00, 0.00] | 2.96 | 0.004

Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed  
 using a Wald t-distribution approximation.

pool.r.squared(m1\_mods)

est lo 95 hi 95 fmi  
R^2 0.09733161 0.01065722 0.2444352 4.081633e-10

## What if we picked model m4?

m4\_mods <- with(ms431\_mice, lm(awake ~ bodywt + brain\_prop + vore))  
m4\_pool <- pool(m4\_mods)  
model\_parameters(m4\_pool, ci = 0.90)

# Fixed Effects  
  
Parameter | Coefficient | SE | 90% CI | t | df | p  
-----------------------------------------------------------------------------------  
(Intercept) | 14.18 | 1.11 | [ 12.33, 16.03] | 12.82 | 63.87 | < .001  
bodywt | 1.50e-03 | 6.04e-04 | [ 0.00, 0.00] | 2.49 | 74.03 | 0.015   
brain prop | -76.36 | 63.63 | [-182.92, 30.20] | -1.20 | 51.74 | 0.236   
vore [herbi] | 0.40 | 1.20 | [ -1.59, 2.40] | 0.34 | 70.78 | 0.737   
vore [insecti] | -4.03 | 2.08 | [ -7.50, -0.56] | -1.94 | 71.22 | 0.057   
vore [omni] | 0.08 | 1.36 | [ -2.18, 2.35] | 0.06 | 68.98 | 0.951

Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed  
 using a Wald t-distribution approximation.

pool.r.squared(m4\_mods)

est lo 95 hi 95 fmi  
R^2 0.1842649 0.0519695 0.3544588 0.06782029

## Guidelines for reporting, I (Sterne)

How should we report on analyses potentially affected by missing data?

* Report the number of missing values for each variable of interest, or the number of cases with complete data for each important component of the analysis. Give reasons for missing values if possible, and indicate how many individuals were excluded because of missing data when reporting the flow of participants through the study. If possible, describe reasons for missing data in terms of other variables (rather than just reporting a universal reason such as treatment failure.)
* Clarify whether there are important differences between individuals with complete and incomplete data, for example, by providing a table comparing the distributions of key exposure and outcome variables in these different groups
* Describe the type of analysis used to account for missing data (e.g., multiple imputation), and the assumptions that were made (e.g., missing at random)

## Guidelines for reporting, II (Sterne)

How should we report on analyses that involve multiple imputation?

* Provide details of the imputation modeling (software used, key settings, number of imputed datasets, variables included in imputation procedure, etc.)
* If a large fraction of the data is imputed, compare observed and imputed values.
* Where possible, provide results from analyses restricted to complete cases, for comparison with results based on multiple imputation. If there are important differences between the results, suggest explanations.
* It is also desirable to investigate the robustness of key inferences to possible departures from the missing at random assumption, by assuming a range of missing not at random mechanisms in sensitivity analyses.

## Session Information

xfun::session\_info()

R version 4.4.1 (2024-06-14 ucrt)  
Platform: x86\_64-w64-mingw32/x64  
Running under: Windows 11 x64 (build 22631)  
  
Locale:  
 LC\_COLLATE=English\_United States.utf8   
 LC\_CTYPE=English\_United States.utf8   
 LC\_MONETARY=English\_United States.utf8  
 LC\_NUMERIC=C   
 LC\_TIME=English\_United States.utf8   
  
Package version:  
 abind\_1.4-8 askpass\_1.2.1 backports\_1.5.0   
 base64enc\_0.1.3 bayestestR\_0.15.0 bigD\_0.3.0   
 bit\_4.5.0 bit64\_4.5.2 bitops\_1.0.9   
 blob\_1.2.4 boot\_1.3-31 broom\_1.0.7   
 bslib\_0.8.0 cachem\_1.1.0 callr\_3.7.6   
 car\_3.1-3 carData\_3.0-5 cellranger\_1.1.0   
 cli\_3.6.3 clipr\_0.8.0 coda\_0.19-4.1   
 codetools\_0.2-20 colorspace\_2.1-1 commonmark\_1.9.2   
 compiler\_4.4.1 conflicted\_1.2.0 correlation\_0.8.6   
 cowplot\_1.1.3 cpp11\_0.5.0 crayon\_1.5.3   
 curl\_6.0.0 data.table\_1.16.2 datasets\_4.4.1   
 datawizard\_0.13.0 DBI\_1.2.3 dbplyr\_2.5.0   
 Deriv\_4.1.6 digest\_0.6.37 doBy\_4.6.24   
 dplyr\_1.1.4 dtplyr\_1.3.1 easystats\_0.7.3   
 effectsize\_0.8.9 emmeans\_1.10.5 estimability\_1.5.1   
 evaluate\_1.0.1 fansi\_1.0.6 farver\_2.1.2   
 fastmap\_1.2.0 fontawesome\_0.5.2 forcats\_1.0.0   
 foreach\_1.5.2 Formula\_1.2-5 fs\_1.6.5   
 gargle\_1.5.2 generics\_0.1.3 GGally\_2.2.1   
 ggformula\_0.12.0 ggplot2\_3.5.1 ggrepel\_0.9.6   
 ggridges\_0.5.6 ggstats\_0.7.0 glmnet\_4.1-8   
 glue\_1.8.0 googledrive\_2.1.1 googlesheets4\_1.1.1   
 graphics\_4.4.1 grDevices\_4.4.1 grid\_4.4.1   
 gridExtra\_2.3 gt\_0.11.1 gtable\_0.3.6   
 haven\_2.5.4 here\_1.0.1 highr\_0.11   
 hms\_1.1.3 htmltools\_0.5.8.1 htmlwidgets\_1.6.4   
 httr\_1.4.7 ids\_1.0.1 insight\_0.20.5   
 isoband\_0.2.7 iterators\_1.0.14 janitor\_2.2.0   
 jomo\_2.7-6 jquerylib\_0.1.4 jsonlite\_1.8.9   
 juicyjuice\_0.1.0 knitr\_1.49 labeling\_0.4.3   
 labelled\_2.13.0 lattice\_0.22-6 lifecycle\_1.0.4   
 lme4\_1.1-35.5 lubridate\_1.9.3 magrittr\_2.0.3   
 markdown\_1.13 MASS\_7.3-61 Matrix\_1.7-0   
 MatrixModels\_0.5.3 memoise\_2.0.1 methods\_4.4.1   
 mgcv\_1.9-1 mice\_3.16.0 microbenchmark\_1.5.0  
 mime\_0.12 minqa\_1.2.8 mitml\_0.4-5   
 modelbased\_0.8.9 modelr\_0.1.11 mosaic\_1.9.1   
 mosaicCore\_0.9.4.0 mosaicData\_0.20.4 multcomp\_1.4-26   
 munsell\_0.5.1 mvtnorm\_1.3-1 naniar\_1.1.0   
 nlme\_3.1-164 nloptr\_2.1.1 nnet\_7.3-19   
 norm\_1.0-11.1 numDeriv\_2016.8.1.1 openssl\_2.2.2   
 ordinal\_2023.12.4.1 pan\_1.9 parallel\_4.4.1   
 parameters\_0.23.0 patchwork\_1.3.0 pbkrtest\_0.5.3   
 performance\_0.12.4 pillar\_1.9.0 pkgconfig\_2.0.3   
 plyr\_1.8.9 prettyunits\_1.2.0 processx\_3.8.4   
 progress\_1.2.3 ps\_1.8.1 purrr\_1.0.2   
 quantreg\_5.99 R6\_2.5.1 ragg\_1.3.3   
 rappdirs\_0.3.3 RColorBrewer\_1.1-3 Rcpp\_1.0.13-1   
 RcppEigen\_0.3.4.0.2 reactable\_0.4.4 reactR\_0.6.1   
 readr\_2.1.5 readxl\_1.4.3 rematch\_2.0.0   
 rematch2\_2.1.2 report\_0.5.9 reprex\_2.1.1   
 rlang\_1.1.4 rmarkdown\_2.29 rpart\_4.1.23   
 rprojroot\_2.0.4 rstudioapi\_0.17.1 rvest\_1.0.4   
 sandwich\_3.1-1 sass\_0.4.9 scales\_1.3.0   
 see\_0.9.0 selectr\_0.4.2 shape\_1.4.6.1   
 snakecase\_0.11.1 SparseM\_1.84.2 splines\_4.4.1   
 stats\_4.4.1 stringi\_1.8.4 stringr\_1.5.1   
 survival\_3.7-0 sys\_3.4.3 systemfonts\_1.1.0   
 textshaping\_0.4.0 TH.data\_1.1-2 tibble\_3.2.1   
 tidyr\_1.3.1 tidyselect\_1.2.1 tidyverse\_2.0.0   
 timechange\_0.3.0 tinytex\_0.54 tools\_4.4.1   
 tzdb\_0.4.0 ucminf\_1.2.2 UpSetR\_1.4.0   
 utf8\_1.2.4 utils\_4.4.1 uuid\_1.2.1   
 V8\_6.0.0 vctrs\_0.6.5 viridis\_0.6.5   
 viridisLite\_0.4.2 visdat\_0.6.0 vroom\_1.6.5   
 withr\_3.0.2 xfun\_0.48 xml2\_1.3.6   
 xtable\_1.8-4 yaml\_2.3.10 zoo\_1.8-12