Carson Slater

6376 Homework 11 Solutions

```
source(here::here("helpers.R"))
```

Preliminary Citations

For inspiration on my linear algebra proofs, I referred back to a paper I wrote in undergrad[1]. As for the work on 2(b), I originally had the intuitive idea that the coefficients were the ratio of two standard normal random variables, so it was Cauchy, but then later, Jacob helped convince me this was true by showing me analytically and I used his solution. For some of the string processing in 4(c) I had ChatGPT 3.5 help me debug some of the code I wrote.

Question 1

(a)

- Positive Definite Matrix: A Hermitian matrix $\mathbf{A}$ is positive definite if for any non-zero column vector in $\mathbb{C}^m$, $\mathbf{x}$, $\mathbf{x}^*\mathbf{A}\mathbf{x} > 0$.
- Positive Semidefinite Matrix: A Hermitian matrix $\mathbf{A}$ is positive semidefinite if for any non-zero column vector in $\mathbb{C}^m$, $\mathbf{x}$, $\mathbf{x}^*\mathbf{A}\mathbf{x} \geq 0$.

(b)

Let $\mathbf{A} \in \mathbb{C}^{m\times m}$, and suppose $\mathbf{A}$ positive definite. Since $\mathbf{A}$ is Hermitian, consider that there exists an eigenpair $(\lambda, \mathbf{v})$ such that $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$, where $\mathbf{v} \neq \mathbf{0}$. Without loss of generality, we assume $\mathbf{v}^*\mathbf{v} = 1$. Then by the definition of a positive definite matrix,

$$0 < \mathbf{v}^*\mathbf{A}\mathbf{v} = \mathbf{v}^*(\lambda\mathbf{v}) = \lambda\mathbf{v}^*\mathbf{v} = \lambda$$

implying all eigenvalues are positive.

Now consider $\mathbf{A} \in \mathbb{C}^{m\times m}$, and suppose $\mathbf{A}$ positive semidefinite. Since $\mathbf{A}$ is Hermitian, consider that there exists an eigenpair $(\lambda, \mathbf{v})$ such that $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$, where $\mathbf{v} \neq \mathbf{0}$. Without loss of generality, we assume $\mathbf{v}^*\mathbf{v} = 1$. Then by the definition of a positive semidefinite matrix,

$$0 \leq \mathbf{v}^*\mathbf{A}\mathbf{v} = \mathbf{v}^*(\lambda\mathbf{v}) = \lambda\mathbf{v}^*\mathbf{v} = \lambda$$

implying all eigenvalues are nonnegative.

[1] You can find the paper I wrote here.

Question 2

(a)

```
roots <- function(coefs) {

  # building Cn
  n <- length(coefs) - 1
  Cn <- rbind(rep(0, n - 1),  diag(1, n - 1))
  Cn <- cbind(Cn, -coefs[1:(n)]/coefs[n + 1])

  # solving Cn
  eigen(Cn)$values
}
```

```
f <- function(x) x^5 - x - 1
coefs <- c(-1, -1, 0, 0, 0, 1)
coefs |> roots() |> round(digits = 3)
```

```
[1]  1.167+0.000i  0.181+1.084i  0.181-1.084i -0.765+0.352i -0.765-0.352i
```

```
coefs |> roots() |> f()
```

```
[1] -1.088019e-14+0.000000e+00i -2.664535e-15-1.110223e-15i
[3] -2.664535e-15+1.110223e-15i  1.776357e-15-1.665335e-16i
[5]  1.776357e-15+1.665335e-16i
```

```
coefs |> roots() |> f() |> is.zero_vec()
```

```
[1] TRUE
```

(b)

Suppose that in selecting a point randomly on the unit circle, we consider $r = 1$ and then $\theta \sim \text{Uniform}(0, 2\pi)$. Let $a_0 = \cos(\theta)$ and $a_1 = \sin(\theta)$. We have that the root of the linear polynomial is $\Re_1 = -\frac{a_0}{a_1} = -\frac{\cos(\theta)}{\sin(\theta)} = -\cot(\theta)$. Then let $g_1(t) = -\cot^{-1}(t)$ for $\theta \in (0, \pi)$, and $g_2(\theta) = -\cot^{-1}(\theta)$ for $\theta \in (\pi, 2\pi)$. So then $g_1^{-1}(t) = -\cot^{-1}(t)$ and $g_2^{-1}(t) = -\cot^{-1}(t) + \pi$

Both g_1 and g_2 are monotonic and $\frac{d}{dt}g_1^{-1}(t) = \frac{d}{dt}g_2^{-1}(t) = \frac{1}{1+t^2}$. By the method of transformations (Casella and Berger Theorem 2.1.8),

$$f_t(t) = f_\theta(g_1^{-1}(t))\left|\frac{d}{dt}g_1^{-1}(t)\right| + f_\theta(g_2^{-1}(t))\left|\frac{d}{dt}g_2^{-1}(t)\right| = \left(\frac{1}{\pi}\right)\frac{1}{1+t^2}.$$

So then $T = -\frac{a_0}{a_1} \sim \text{Cauchy}(0, 1)$.

Question 3

(a)

The matrix $\mathbf{S}_n \in \mathbb{R}^{n\times n}$ is the cumulative sum operation on vector $\mathbf{x} \in \mathbb{R}^n$. Each element in vector $\mathbf{x}$ is mapped to be the sum of itself with the sum of every preceding element in the original vector.

(b)

```
Sn <- function(n) {
  # create I_n
  mat <- diag(n)

  # fill in lower values
  for (i in 1:n) {
    for (j in 1:i) {
        mat[i, j] <- 1  # Fill lower diagonal with 1's
    }
  }
  mat
}
```

```
Sn(4)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    1    1    0    0
[3,]    1    1    1    0
[4,]    1    1    1    1
```

(c)

```
mesh <- function(a, b, n, inner = TRUE) {

  vec <- rep(a, times = n)
  index <- 1:n

  # shift for inner
  if (inner) {
    h <- (b - a) / n
    vec <- imap(vec, .f = \(x, index) x + h/2 + (h*(index - 1))) |> unlist()
  } else {
    # outer default
    h <- (b - a) / (n - 1)
    vec <- imap(vec, .f = \(x, index) x + h*(index - 1)) |> unlist()
  }

  # attributes
  attr(vec, "h") <- h
  attr(vec, "a") <- a
  attr(vec, "b") <- b

  vec
}
```

```
mesh(0, 1, 3)
```

```
[1] 0.1666667 0.5000000 0.8333333
attr(,"h")
[1] 0.3333333
attr(,"a")
[1] 0
attr(,"b")
[1] 1
```

```
mesh(0, 1, 3, inner = FALSE)
```

```
[1] 0.0 0.5 1.0
attr(,"h")
[1] 0.5
```

```
attr(,"a")
[1] 0
attr(,"b")
[1] 1
```

(d)

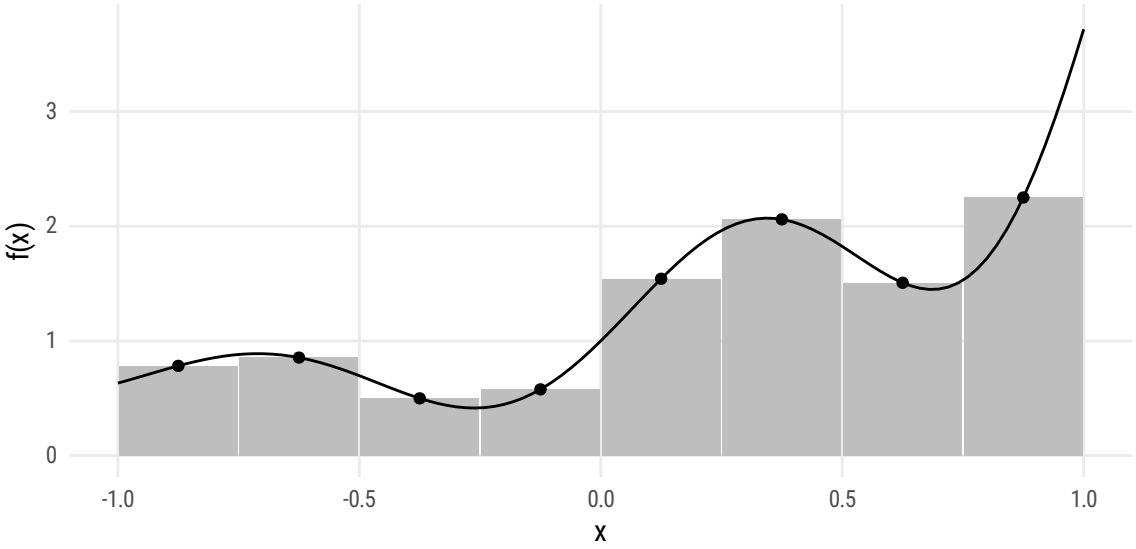
```r
sample_f <- function(f, a ,b, n, inner = TRUE) {

  # check for function
  stopifnot(is.function(f))

  m <- mesh(a, b, n, inner = inner)
  tibble(
    "x"  = m,
    "fx" = f(m),
    "a"  = rep(a, times = n),
    "b"  = rep(b, times = n),
    "h"  = rep(attr(m, "h"), times = n)
    )
}
```

```r
f <- function(x) exp(x)*(x + sin(2*pi*x)/2) + 1
n <- 8
(df <- sample_f(f, -1, 1, n))
```

```
# A tibble: 8 x 5
       x    fx     a     b     h
   <dbl> <dbl> <dbl> <dbl> <dbl>
1 -0.875 0.783    -1     1  0.25
2 -0.625 0.855    -1     1  0.25
3 -0.375 0.499    -1     1  0.25
4 -0.125 0.578    -1     1  0.25
5  0.125 1.54     -1     1  0.25
6  0.375 2.06     -1     1  0.25
7  0.625 1.51     -1     1  0.25
8  0.875 2.25     -1     1  0.25
```

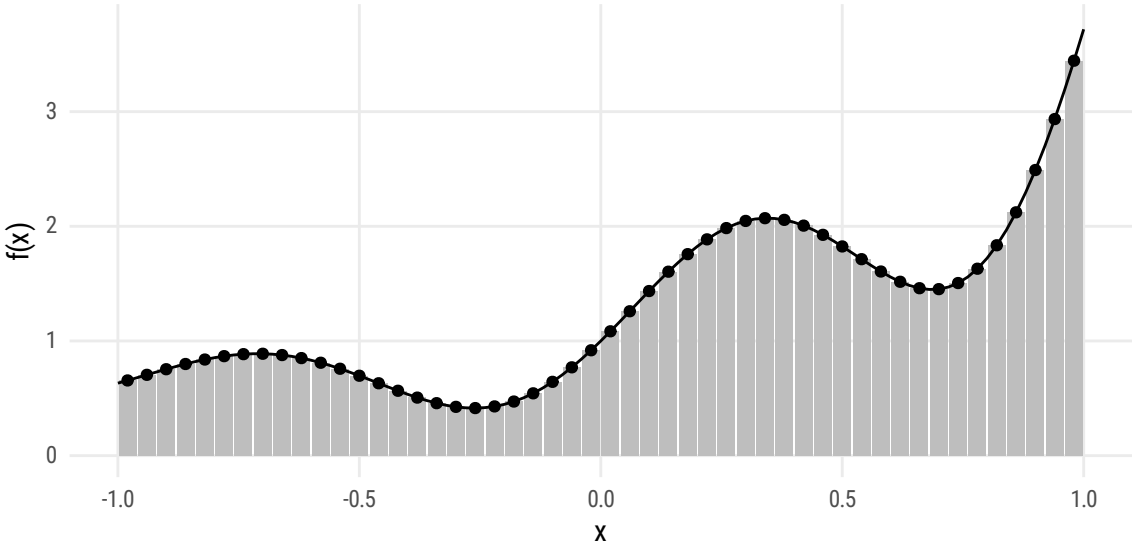
(e)

```
df <- df |>
  mutate(
    'xmin' = x - h/2 + 0.002,
    'xmax' = x + h/2 - 0.002,
    'ymin' = 0,
    'ymax' = fx
    )

df |> ggplot() +
  geom_rect(aes(xmin = xmin,
                xmax = xmax,
                ymin = ymin,
                ymax = ymax),
            fill = "grey") +
  geom_function(fun = f, xlim = c(-1, 1)) +
  geom_point(aes(x, fx)) +
  ylim(0, 3.75) +
  labs(y = "f(x)")
```



(f)

```r
n <- 50
df2 <- sample_f(f, -1, 1, n) |>
  mutate(
    'xmin' = x - h/2 + 0.002,
    'xmax' = x + h/2 - 0.002,
    'ymin' = 0,
    'ymax' = fx
    )

df2 |> ggplot() +
  geom_rect(aes(xmin = xmin,
                xmax = xmax,
                ymin = ymin,
                ymax = ymax),
            fill = "grey") +
  geom_function(fun = f, xlim = c(-1, 1)) +
  geom_point(aes(x, fx)) +
  ylim(0, 3.75) +
  labs(y = "f(x)")
```



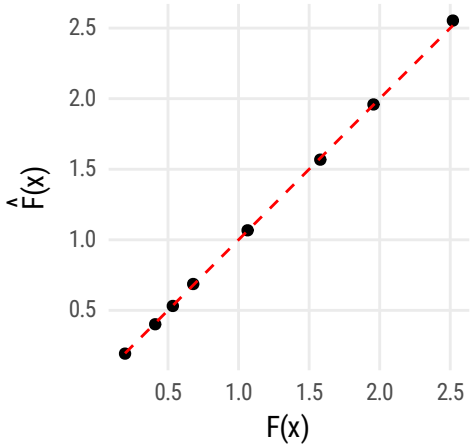
(g)

```r
n <- 8
m <- mesh(-1, 1, n, inner = TRUE) + attributes(mesh(-1, 1, n, inner = TRUE))$h/2
```

```
accuracy_df <- tibble(
    'x'    = m,
    'Fhat' = attributes(m)$h * Sn(n) %*% f(df$x) |> as.vector(),
    'F'    = sapply(m, \(m) integrate(f, -1, m)$value)
    ) |>
  dplyr::select(x, Fhat, `F`)
```

(h)

```
accuracy_df |> ggplot() +
  geom_point(aes(Fhat, `F`)) +
  geom_function(fun = \(.) ., color = "red", linetype = 2) +
  coord_equal() +
  labs(x = latex2exp::TeX(r'(F(x))'),
       y = latex2exp::TeX(r'(\hat{F}(x))'))
```



(i)

$\mathbf{S}_n^{-1}$ is an $n \times n$ matrix with 1's on the diagonal, and for each entry below each diagonal entry, there is a -1. For $(h\mathbf{S}_n)^{-1}$, each value on the diagonal is $\frac{1}{h}$, where each value below each diagonal value is $-\frac{1}{h}$.

```
iSn <- function(n) {

  # create diagonal
```

```
  mat <- diag(n)

  # create second diagonal
  mat2 <- (-1)*diag(n-1) |> cbind(numeric(n-1))
  mat2 <- rbind(numeric(n), mat2)

  mat + mat2
}
```

```
iSn(10) %*% Sn(10)
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1    0    0    0    0    0    0    0    0     0
 [2,]    0    1    0    0    0    0    0    0    0     0
 [3,]    0    0    1    0    0    0    0    0    0     0
 [4,]    0    0    0    1    0    0    0    0    0     0
 [5,]    0    0    0    0    1    0    0    0    0     0
 [6,]    0    0    0    0    0    1    0    0    0     0
 [7,]    0    0    0    0    0    0    1    0    0     0
 [8,]    0    0    0    0    0    0    0    1    0     0
 [9,]    0    0    0    0    0    0    0    0    1     0
[10,]    0    0    0    0    0    0    0    0    0     1
```

(j)

The $(h\mathbf{S}_n)^{-1}$ operation is the differentiation operator. Consider the i$^{\text{th}}$ row of $(h\mathbf{S}_n)^{-1}$, say $ha_i = (0, \dots, 0, -\frac{1}{h}, \frac{1}{h}, 0, \dots, 0)$. Then $\langle ha_i, \mathbf{F} \rangle = 0 + \dots + 0 + (-\frac{1}{h})f_{i-1} + (\frac{1}{h})f_i + 0 + \dots + 0 = \frac{f(x_i+h)-f(x_i)}{h}$.

(k)

The matrices $(h\mathbf{S}_n)^2$ and $(h\mathbf{S}_n)^{-2}$ are the double integration and second differentiation operators, respectively.

```
Sn(5) %*% Sn(5)
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    2    1    0    0    0
```

```
[3,]    3    2    1    0    0
[4,]    4    3    2    1    0
[5,]    5    4    3    2    1
```

```
iSn(5) %*% iSn(5)
```

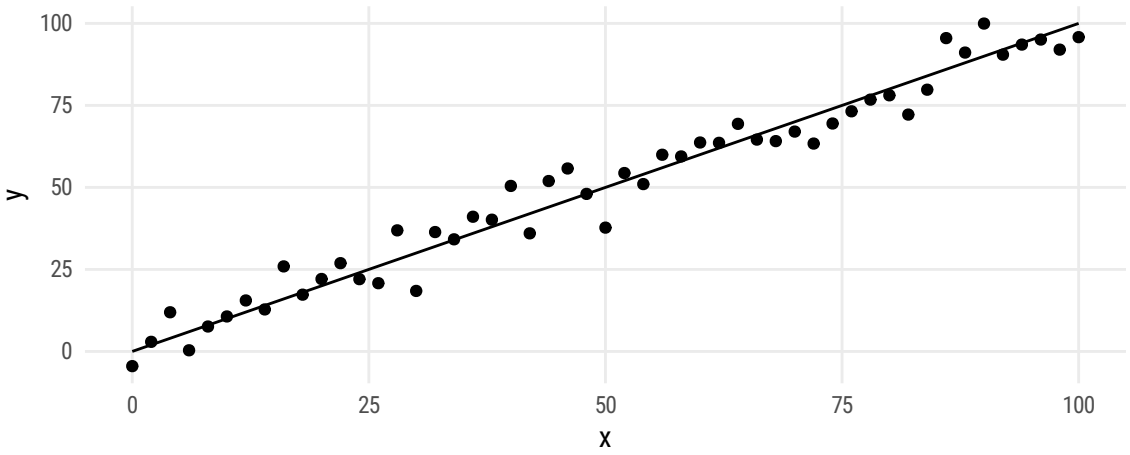
```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]   -2    1    0    0    0
[3,]    1   -2    1    0    0
[4,]    0    1   -2    1    0
[5,]    0    0    1   -2    1
```

Question 4

(a)

```
set.seed(2L)

be0 <- 0
be1 <- 1
sig_sq <- 5^2

tb <- tibble(
  x = seq(0, 100, length.out = 51),
  y = be0 + be1*x + rnorm(51, sd = sqrt(sig_sq))
)
```

```
ggplot(tb, aes(x, y)) +
  geom_point() +
  stat_function(fun = ~ be0 + be1*.x)
```



(b)

```
pwr <- -2:2
be1 <- map_vec(pwr, \(k) 100*2^(k))
sig <- seq(1000, 2500, 500)

gen_data <- function(be1, sig, ...) {
  tb <- tibble(
    x = seq(0, 100, length.out = 51),
    # be0 is always zero
    y = be1*x + rnorm(51, sd = sig)
  )
  write_csv(tb, file = path("data", paste0("be1=", be1, "-si=", sig,".csv")))
}

if (!dir_exists("data")) dir_create("data")

for (i in seq_along(be1)) {
  for (j in seq_along(sig)) {
    gen_data(be1[i], sig[j])
  }
}
```

```
grab <- function(x, i) x[i]
my_files <- "data" |> dir_ls() |> path_file()
my_files |> grab(1:6)
```

```
[1] "be1=100-si=1000.csv" "be1=100-si=1500.csv" "be1=100-si=2000.csv"
[4] "be1=100-si=2500.csv" "be1=200-si=1000.csv" "be1=200-si=1500.csv"
```

(c)

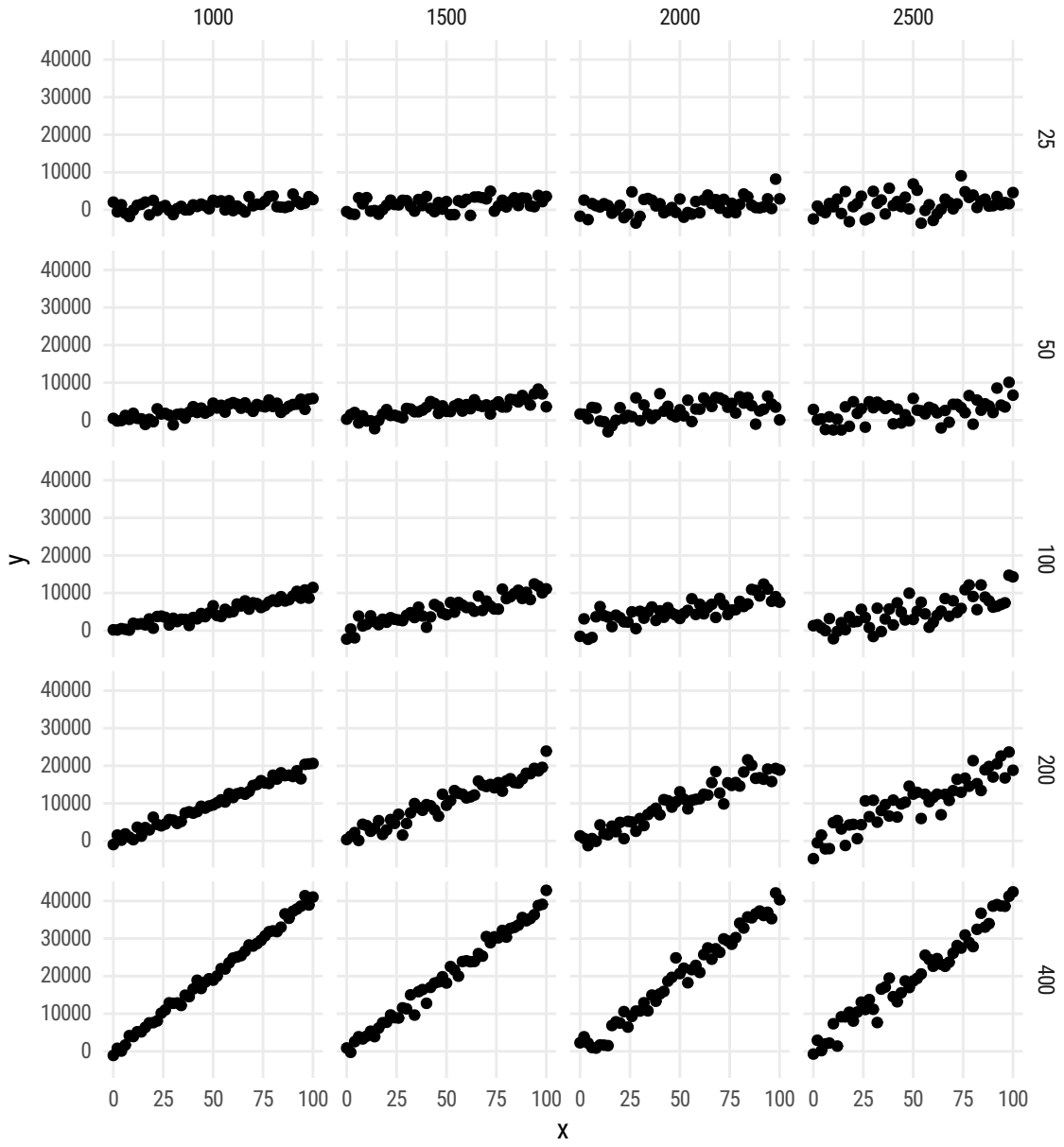
```
beta1 <- my_files |> str_extract_all("([[:digit:]]+)") |> sapply(\(x) x[2])

sigma <-  my_files |> str_extract_all("([[:digit:]]+)") |> sapply(\(x) x[3])

read_and_clean_file <- function(file, be1, sig) {
  df <- read_csv(paste0("data/", file),
                 col_types = list("n", "n")
                 )
  df <- df |>
    mutate(
      be1 = rep(be1, length = nrow(df)),
      si = rep(sig, length = nrow(df))
    )
}

datasets <- list()

for (i in seq_along(my_files)) {
  datasets[[i]] <- read_and_clean_file(my_files[i], beta1[i], sigma[i]) |>
    suppressWarnings()
}

names(datasets) <- my_files

datasets |> grab(1:3) |> str(give.attr = FALSE)
```

```
List of 3
 $ be1=100-si=1000.csv: tibble [51 x 4] (S3: tbl_df/tbl/data.frame)
  ..$ x  : num [1:51] 0 2 4 6 8 10 12 14 16 18 ...
  ..$ y  : num [1:51] 204 163 516 344 125 ...
  ..$ be1: chr [1:51] "100" "100" "100" "100" ...
  ..$ si : chr [1:51] "1000" "1000" "1000" "1000" ...
 $ be1=100-si=1500.csv: tibble [51 x 4] (S3: tbl_df/tbl/data.frame)
  ..$ x  : num [1:51] 0 2 4 6 8 10 12 14 16 18 ...
  ..$ y  : num [1:51] -2254 413 -1935 3865 1243 ...
```

```
  ..$ be1: chr [1:51] "100" "100" "100" "100" ...
  ..$ si : chr [1:51] "1500" "1500" "1500" "1500" ...
 $ be1=100-si=2000.csv: tibble [51 x 4] (S3: tbl_df/tbl/data.frame)
  ..$ x  : num [1:51] 0 2 4 6 8 10 12 14 16 18 ...
  ..$ y  : num [1:51] -1529 3123 -2308 -1836 3677 ...
  ..$ be1: chr [1:51] "100" "100" "100" "100" ...
  ..$ si : chr [1:51] "2000" "2000" "2000" "2000" ...
```

(d)

```
data <- datasets |>
  reduce(rbind) |>
  mutate(
    be1 = factor(be1, order = TRUE,
                 levels = c("25", "50", "100", "200", "400")),
    si = factor(si, order = TRUE,
                levels = c("1000", "1500", "2000", "2500"))
  )

data |> ggplot(aes(x, y)) +
  geom_point() +
  facet_grid(rows = vars(be1), cols = vars(si))
```



(e)

```
walk(my_files, \(file) file_delete(paste0("data/", file)))
dir_ls("data")
```

```
character(0)
```