Ho, ho, ho, Happy Chris.. New Year? Between eating the sea of fish (as the [Polish tradition requires](https://en.wikipedia.org/wiki/Christmas_in_Poland)), assembling doll houses and designing a new kitchen, I finally managed to publish the third post on [My R take on Advent of Code](https://r-tastic.co.uk/post/advent-of-code-in-r-day-1/). To keep things short and sweet, here’s the original challenge:

Each Elf has made a claim about which area of fabric would be ideal for Santa’s suit. All claims have an ID and consist of a single rectangle with edges parallel to the edges of the fabric. Each claim’s rectangle is defined as follows:  
The number of inches between the left edge of the fabric and the left edge of the rectangle.  
The number of inches between the top edge of the fabric and the top edge of the rectangle.  
The width of the rectangle in inches.  
The height of the rectangle in inches.  
A claim like #123 @ 3,2: 5×4 means that claim ID 123 specifies a rectangle 3 inches from the left edge, 2 inches from the top edge, 5 inches wide, and 4 inches tall. Visually, it claims the square inches of fabric represented by # (and ignores the square inches of fabric represented by .) in the diagram below:

………..  
………..  
…#####…  
…#####…  
…#####…  
…#####…  
………..  
………..  
………..

The problem is that many of the claims overlap, causing two or more claims to cover part of the same areas. For example, consider the following claims:

#1 @ 1,3: 4×4  
#2 @ 3,1: 4×4  
#3 @ 5,5: 2×2

Visually, these claim the following areas:

……..  
…2222.  
…2222.  
.11XX22.  
.11XX22.  
.111133.  
.111133.  
……..

The four square inches marked with X are claimed by both 1 and 2. (Claim 3, while adjacent to the others, does not overlap either of them.). If the Elves all proceed with their own plans, none of them will have enough fabric. How many square inches of fabric are within two or more claims?

This is interesting! let’s load tidyverse and have a quick look at the data:

library(tidyverse)

raw\_input <- read.delim('day3-raw-input.txt', header = FALSE)

head(raw\_input)

## V1

## 1 #1 @ 850,301: 23x12

## 2 #2 @ 898,245: 15x10

## 3 #3 @ 8,408: 12x27

## 4 #4 @ 532,184: 16x13

## 5 #5 @ 550,829: 11x10

## 6 #6 @ 656,906: 13x12

Ha! It looks like we need to first extract each dimension from the original input – easy-peasy with a little bit of regex and parse\_number:

# separate and clean dimension figures

clean\_input <- raw\_input %>%

rename(input = V1) %>%

mutate(ID = str\_extract(input, '#[:digit:]+'), # extract ID

from\_left\_edge = str\_extract(input, '@..?[:digit:]+\\,'), # extract right squares

from\_top\_endge = str\_extract(input, '\\,[:digit:]+\\:'), # extract top square

width = str\_extract(input, '[:digit:]+x'), # extract left dimension

height = str\_extract(input, 'x[:digit:]+')# extract right dimension

) %>%

mutate\_if(is.character, readr::parse\_number) # extract numbers

head(clean\_input, 10)

## input ID from\_left\_edge from\_top\_endge width height

## 1 #1 @ 850,301: 23x12 1 850 301 23 12

## 2 #2 @ 898,245: 15x10 2 898 245 15 10

## 3 #3 @ 8,408: 12x27 3 8 408 12 27

## 4 #4 @ 532,184: 16x13 4 532 184 16 13

## 5 #5 @ 550,829: 11x10 5 550 829 11 10

## 6 #6 @ 656,906: 13x12 6 656 906 13 12

## 7 #7 @ 489,357: 24x23 7 489 357 24 23

## 8 #8 @ 529,898: 12x19 8 529 898 12 19

## 9 #9 @ 660,201: 19x28 9 660 201 19 28

## 10 #10 @ 524,14: 21x27 10 524 14 21 27

Now that we have the dimensions, how do we go about determining the overlap?  
My idea was to, for each claim, create a series of ‘coordinates’ where the digit before a dot indicates the position from the left edge and the second number the position from the top edge. Overlapping squares from different samples would have the same ‘coordinates’.

Right, but how to do it in R? this is where outer function comes in handy. Let’s take the first example:

# first example

#1 @ 1,3: 4x4

from\_left\_edge <- 1

width <- 4

from\_top\_endge <- 3

height <- 4

# create a series of coordinates per set of dimensions

dims <- as.vector(outer(from\_left\_edge + 1:width,

from\_top\_endge + 1:height,

paste, sep = '.'))

sort(dims)

## [1] "2.4" "2.5" "2.6" "2.7" "3.4" "3.5" "3.6" "3.7" "4.4" "4.5" "4.6"

## [12] "4.7" "5.4" "5.5" "5.6" "5.7"

Neat! Now, let’s wrap it up in a function and apply it to the challenge dataset:

#function that creates coordicates for squares occupied by each claim

get\_dimensions <- function(from\_left\_edge,

width,

from\_top\_endge,

height

) {

dims <- as.vector(outer(from\_left\_edge + 1:width,

from\_top\_endge + 1:height,

paste, sep = '.'))

return(dims)

}

## apply the function to the challenge dataset

final\_list <- pmap(list(from\_left\_edge = clean\_input$from\_left\_edge,

width = clean\_input$width,

from\_top\_endge = clean\_input$from\_top\_endge,

height = clean\_input$height),

get\_dimensions)

# a list of 'coordinates' for the first claim

head(final\_list[[1]])

## [1] "851.302" "852.302" "853.302" "854.302" "855.302" "856.302"

Now, let’s calculate the number of those coordiates that appear more than once in our list, which will give us the final solution to the Day 3 Puzzle:

## final solution

final\_list%>%

unlist() %>%

table() %>% # get counts per coordinate

as\_tibble() %>% # put it in usable format

filter(n > 1) %>%

nrow()

## [1] 113576