Manipulating Data

Load packages

Use the readr, tidyr and dplyr from Tidyverse. Load in all the packages

```
library(readr)
library(tidyr)
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(here)
```

```
## here() starts at C:/Users/romin/ToyRepo
```

Practice with pivoting

Pivoting is described as a summary of the data, based on a certain category. You can pivot large datasets to provide a summary. The practical starts with a non tidy dataset

An important note is that for a dataset to be tidy each row must contain only one observation, in this dataset there is 3 observations per row. a1,b1 and c1 are all contained within one row

```
patientID <- c(1,2)
test_result_month1 <- c("a1","a2")
test_result_month2 <- c("b1","b2")
test_result_month3 <- c("c1","c2")

patient_test <- data.frame(patientID, test_result_month1,test_result_month2,test_result_month3)
patient_test
```

```
##   patientID test_result_month1 test_result_month2 test_result_month3
## 1         1                 a1                 b1                 c1
## 2         2                 a2                 b2                 c2
```

- The first thing we have to do is reshape the dataset, %>% is a pipe function that will lead one function into the other using the result of the previous function. we use the result of the first argument to pipe into the other function. We use the pivot longer function, this function will take a dataset that is wide in size and change it into something long. In this case we go from 3 observations in a row to 1, to do this we need to change the shape of the dataset. It does this by the user defining, new character and value columns

```
tidy_patient_tests<- patient_test %>%
    pivot_longer(
```

```
        c('test_result_month1','test_result_month2','test_result_month3'),
        names_to='month',
        values_to='test_result'
    )
tidy_patient_tests
```

```
## # A tibble: 6 x 3
##   patientID month              test_result
##       <dbl> <chr>              <chr>
## 1         1 test_result_month1 a1
## 2         1 test_result_month2 b1
## 3         1 test_result_month3 c1
## 4         2 test_result_month1 a2
## 5         2 test_result_month2 b2
## 6         2 test_result_month3 c2
```

To make the overall document look cleaner we can remove the long variable name and add a small prefix to it, to replace the variable name. The month value replaces all text. It takes all the text that is within

```
tidy_patient_tests <- patient_test %>%
  pivot_longer(
    c('test_result_month1','test_result_month2','test_result_month3'),
    names_to= 'month',
    names_prefix = 'test_result_month',
    names_transform = list(month = as.integer),
    values_to='test_result'
    )
tidy_patient_tests
```

```
## # A tibble: 6 x 3
##   patientID month test_result
##       <dbl> <int> <chr>
## 1         1     1 a1
## 2         1     2 b1
## 3         1     3 c1
## 4         2     1 a2
## 5         2     2 b2
## 6         2     3 c2
```

Reading in FEV data

Just read in the values for the FEV data:

```
fev_data <- read_csv("C:\\Users\\romin\\ToyRepo\\fev.csv")
```

```
## Rows: 654 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (7): seqnbr, subjid, age, fev, height, sex, smoke
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Using the $ symbol we can extract certain qualities about the data. The $ can have many functions within the code, but the one used here is to select a paticular item in the code. This result will be saved as a vector form, containing information about the data.

```r
fev_data$fev[32] # Returns the 32nd element from the fev column
```

```
## [1] 3
```

```r
fev_data[32,3] # Returns the 32nd element of the 3rd column in fev
```

```
## # A tibble: 1 x 1
##     age
##   <dbl>
## 1     9
```

```r
fev_data[32, "age"] # Same as above, it just makes it more readable. 3 = "age" column
```

```
## # A tibble: 1 x 1
##     age
##   <dbl>
## 1     9
```

```r
fev_data[32, ] #Everything in the 32nd element of fev column is returned
```

```
## # A tibble: 1 x 7
##   seqnbr subjid   age   fev height   sex smoke
##    <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1     32   7201     9     3   65.5     1     0
```

```r
fev_data[32, 1:3] # Returns 32nd element from the 1st to the 3rd column
```

```
## # A tibble: 1 x 3
##   seqnbr subjid   age
##    <dbl>  <dbl> <dbl>
## 1     32   7201     9
```

```r
fev_data[32, -5] # This means that the index value 5 will be exculded from the vector
```

```
## # A tibble: 1 x 6
##   seqnbr subjid   age   fev   sex smoke
##    <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     32   7201     9     3     1     0
```

```r
fev_data[32, -1:-2] # This excludes a slice of the dataset in the result
```

```
## # A tibble: 1 x 5
##     age   fev height   sex smoke
##   <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1     9     3   65.5     1     0
```

To drop the 1st, 3rd and 5th column we do the following:

```r
fev_data[c(-1,-3,-5)]
```

```
## # A tibble: 654 x 4
##    subjid   fev   sex smoke
##     <dbl> <dbl> <dbl> <dbl>
##  1    301  1.71     0     0
##  2    451  1.72     0     0
##  3    501  1.72     0     0
##  4    642  1.56     1     0
##  5    901  1.90     1     0
##  6   1701  2.34     0     0
```

```
##  7   1752  1.92     0     0
##  8   1753  1.42     0     0
##  9   1901  1.99     0     0
## 10   1951  1.94     0     0
## # i 644 more rows
```

Logicals

Logicals such as TRUE or FALSE values can also be held in a vector. You can also set conditions to them such as restricting height

```
is_tall <- fev_data$height > 72 # Takes the height column and restricts the data to values where height
```

To print these values we can use a summary table

```
table(is_tall)
```

```
## is_tall
## FALSE  TRUE
##   647     7
```

To be more specific/detailed of what the values actually result in we can pass the condition inside of the dataframe vector

```
fev_data[is_tall,]
```

```
## # A tibble: 7 x 7
##   seqnbr subjid   age   fev height   sex smoke
##    <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1    401  18841    14  4.27   72.5     1     0
## 2    450  32741    13  4.22   74       1     0
## 3    464  37241    13  4.88   73       1     0
## 4    517  49541    13  5.08   74       1     0
## 5    550  59941    14  4.27   72.5     1     0
## 6    632  37441    17  5.63   73       1     0
## 7    636  44241    16  3.64   73.5     1     0
```