Data Manipulation

Master the Data Wrangle

Studies have shown that data analysts spend up to 80 percent of their time preparing data for analysis.

This type of work involves creating new variables and summary statistics, merging data sets, and so forth.

Fortunately R offers many functions for this so-called "data wrangling." We have touched on some of these functions in the previous section. In this section we will focus on some features of the `dplyr` library.

Engaging the tidyverse

This session takes advantage of many of the features of the `dplyr`, a library in the `tidyverse`. You already installed it when running `install.packages("tidyverse")`. Let's load it.

```
library(tidyverse)
```

Reading in our data

For this section we are going to read in the `hsbraw.csv()` dataset, taken from the Institute for Digital Research and Education at UCLA. This section is largely based on their R seminar, a good resource for after this seminar.

```
d<-read_csv("C:/RFiles/hsbraw.csv")
```

Variable Transformation

Often we need to create variables from other variables. For example, we may want to sum individual test items to form a total score. Or, we may want to convert a continuous scale into several categories, such as letter grades.

Some useful functions to transform variables:

- `log()`: logarithm
- `min_rank()`: rank values
- `scale()`: standardizes variable (substracts mean and divides by standard deviation)
- `rowMeans()`, `rowSums()`: means and sums of several columns

Adding new variables to the data frame

You can add variables to data frames by declaring them as column variables of that data frame.

```
# this will add a column named zmath which is a standardize
d$zmath <- log(d$math)
head(d$zmath,10)
##  [1] 3.713572 3.713572 3.784190 3.737670 3.688879 3.7376
##  [8] 3.688879 3.496508 3.828641
```

Now we see that zmath is another column in the d data frame.

```
names(d)
##  [1] "id"      "female"  "ses"     "schtyp"  "prog"    "
##  [8] "math"    "science" "socst"   "honors"  "awards"  "
```

Tranforming many variables at once with mutate()

The dplyr function mutate() allows us to transform many variables in one step without having to respecify the data frame name over and over.

Below we transform math in 4 different ways.

```
# create 4 transformations of the read variable
d <- mutate(d,
            logmath = log(read),
            mathrank = min_rank(read),
            zmath = scale(read)
            )
names(d)
##  [1] "id"       "female"   "ses"      "schtyp"   "prog"
##  [7] "write"    "math"     "science"  "socst"    "honors
## [13] "cid"      "zmath"    "logmath"  "mathrank"
```

Subsetting rows of a data frame with filter()

The dplyr function `filter()` provides a cleaner syntax for subsetting datasets.

```
# subset to females with high math
d_fem_hi_math <- filter(d, female == "female" & math > 50)
dim(d_fem_hi_math)
## [1] 62 16
```

```
# subset to students with math < 50 in the general or acad
d_gen_aca_low_math <- filter(d,
        (prog == "general" | prog == "academic") & math < 5
dim(d_gen_aca_low_math)
## [1] 47 16
```

Appending observations (appending by rows)

Sometimes we are given our dataset in parts, with observations spread over many files (collected by different researchers, for example). To create one dataset, we need to append the datasets together row-wise.

The function `rbind()` appends data frames together. The variables must be the same between datasets.

Here, we `rbind()` the two datasets we created with `filter()` above, and check that it was successful by calculating the number of rows.

```
# rbind works because they have the same variables
d_append <- rbind(d_fem_hi_math, d_gen_aca_low_math)

# dimensions of component datasets
dim(d_fem_hi_math)
## [1] 62 16
dim(d_gen_aca_low_math)
## [1] 47 16



# appended dataset has rows = sum of rows of components
dim(d_append)
## [1] 109  16
```

Subsetting Variables (columns)

Often, datasets come with many more variable than we want. We can use the dplyr function `select()` to keep only the variables we need.

```
# select 4 variables
d_use <- select(d, id, female, read, write)
names(d_use)
## [1] "id"     "female" "read"   "write"
```

```
# select everything BUT female, read, write
# note the - preceding c(female...)
d_dropped <- select(d, -c(female, read, write))
names(d_dropped)
##  [1] "id"       "ses"      "schtyp"   "prog"      "math"
##  [7] "socst"    "honors"   "awards"   "cid"       "zmath"
## [13] "mathrank"
```

Combining columns of data

If we know that the rows of data of 2 columns (or two data frames) correspond to the same observations, we can use cbind() to combine the columns into a single data frame. Columns combined this way must have the same number of rows.

The rows of the two data frames we just created with select() indeed do correspond to the same observations:

```
d_all <- cbind(d_use, d_dropped)
names(d_all)
##  [1] "id"       "female"   "read"     "write"    "id"
##  [7] "schtyp"   "prog"     "math"     "science"  "socst"
## [13] "awards"   "cid"      "zmath"    "logmath"  "mathra
```

Adding data columns by merging on a key variable

More often, we receive separate datasets with different variables (columns) that must be merged on a key variable.

Merging is an involved topic, with many different kinds of merges possible. We will solely demonstrate merges where only matched observations are kept.

Challenge question: What is the median math score for each student's class?

groupby()

This exercise takes advantage of multiple `dplyr` functions. First, we will create a *grouped* data frame, grouping the data by class. This will allow us to perform operations on each *group*, i.e. calculate each group's median.

We do this with the `group_by` function - grouping our tibble `d` by class ID `cid`.

```
# first group data by cid (there are 20 classes)
by_class <- group_by(d, cid)
```

Summarize()

Now that we have our data grouped, we can perform operations on each group. `Summarize()` allows us to calculate summary statistics *for each individual group* and place that as a column in our tibble.

```
# then get mean/median on math by class
class_median <- summarize(by_class,medmath=median(math))
names(class_median)
## [1] "cid"     "medmath"
```

And now, the merge.

Right now, we have two tibbles - one with the median math score for each class, and one with all the other data. We want to merge these two datasets.

We will use the dplyr function `inner_join()` to merge the datasets. This will search both datasets for any variables with the same name, and will use those as matching variables. If you need to control which variables are used to match, use the by= argument.

In our two datasets, the only variable that appears in both is cid, which we want to use as the key variable, so we do not need by=:

```
## Joining, by = "cid"
d_merged <- inner_join(d, median)
head(d_merged)
```

DRILLS

1. Create a new dataframe, `irislog`, based on the existing `iris` dataframe.
2. Create columns `Petal.Length.Log` and `Petal.Width.Log` which are log-transformations of their respective fields.
3. Filter this dataframe to contain only records from the `setosa` species.
4. Sub-set this dataframe to keep only the `Species`, `Petal.Length.Log` and `Petal.Width.Log` fields.

Questions?