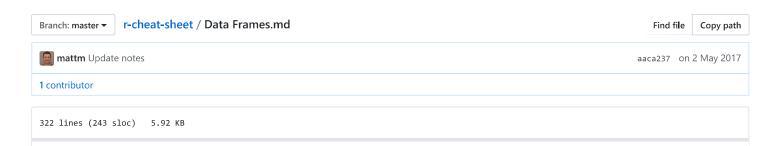
#### mattm / r-cheat-sheet



## **Data Frames**

A typical data set contains data of different modes. In an employee data set, for example, we might have character string data, such as employee names, and numeric data, such as salaries. So, although a data set of (say) 50 employees with 4 variables per worker has the look and feel of a 50-by-4 matrix, it does not qualify as such in R, because it mixes types.

Instead of a matrix, we use a data frame. A data frame in R is a list, with each component of the list being a vector corresponding to a column in our "matrix" of data. Indeed, you can create data frames in just this way:

The Art of R Programming

#### Creating a data frame from lists

## Checking the dimensions of a data frame

That's rows then columns:

### Returning the column names

```
> colnames(d)
[1] "kids" "ages"
```

## Viewing a summary of the data

### Viewing the structure of the data

```
> str(d)
'data.frame': 3 obs. of 2 variables:
$ kids: Factor w/ 3 levels "Jack","Jill",..: 1 2 3
$ ages: num 12 10 4
```

### Returning the values of a data frame component

Just like you would a list:

```
> d$kids
[1] Jack Jill
Levels: Jack Jill
```

#### Returning a component of the data frame

Use single brackets [] to return a list:

```
> d['kids']
  kids
1 Jack
2 Jill
```

## Using the standard [] method

```
> d <- data.frame(list(kids = c("Jack", "Jill"), ages = c(12, 10)))
> d[d$kids == "Jack",]
   kids ages
1 Jack 12
```

## Subsetting using subset

#### Single column, exact value

```
> fl = filter(housing, State == "FL")
```

#### Single column, any of multiple values

```
With subset:
  both = subset(housing, State %in% c("FL", "GA"))
With dplyr's filter:
  > both = filter(housing, State == "FL" | State == "GA")
```

#### Multiple columns

### Re-ordering rows

```
With order:

> # Ascending
> housing[order(housing$Home.Value), ]

> # Descending
> housing[order(-housing$Home.Value), ]
> housing[order(housing$Home.Value, decreasing = TRUE), ]

With dplyr's arrange:

> # Ascending
> arrange(housing, Home.Value)

> # Descending
> arrange(housing, desc(Home.Value))
```

# Selecting columns

### Removing a column

```
With select:
   housing <- select(housing, -State)</pre>
```

### Renaming a column

Note that State. Name is the new name.

### Extract distinct (unique) rows

```
With unique:
    > unique(housing[, c("State", "region")])
With dplyr's distinct:
    > distinct(housing, State, region)
```

# Removing NA values

```
Use complete.cases:

> d <- data.frame(list(kids = c("Jack", "Jill"), ages = c(12, NA)))
> d
    kids ages
1 Jack    12
2 Jill    NA
> d[complete.cases(d), ]
    kids ages
1 Jack    12
```

## Taking a sample

#### Adding a new column

```
> d <- data.frame(list(name = c("Jack", "Jill"), age = c(12, 10)))</pre>
  > d
   name age
  1 Jack 12
  2 Jill 10
Natively:
  > d$next_age = d$ages + 1
  > d
   name age next_age
  1 Jack 12 13
  2 Jill 10
                   11
With dplyr's mutate:
  > d <- mutate(d, next_age = ages + 1)</pre>
  > d
   name age next_age
  1 Jack 12 13
  2 Jill 10
                  11
Note that with mutate, you can reference columns you're currently adding:
  > d <- mutate(d, next_age = age + 1, next_next_age = next_age + 1)</pre>
```

### **Grouping Operations**

name age next\_age next\_next\_age
1 Jack 12 13 14

11

> d

2 Jill 10

#### Applying summarize to groups of observations

With one summary statistic:

With multiple:

```
4 AZ 153 140755.59
5 CA 153 282808.08
```

Note that n() is an aggregate function provided by dplyr that returns the number of observations in each group, which in this example are all 153.

There are other aggregate functions as well:  $n_{distinct(x)}$  (the number of unique values of x), first(x), last(x), and nth(x).

We can chain dplyr functions together using the %>% operator:

```
> group_by(housing, State) %>% summarize(Avg.Home.Value = mean(Home.Value))
# A tibble: 51 \times 2
   State Avg.Home.Value
  <fctr>
                 <dbl>
          147385.14
1
     AK
            92545.22
2
     AL
            82076.84
3
     AR
     AZ 140755.59
4
     CA
            282808.08
```