Working with Data in Julia

John Myles White

January 14, 2013

```
Pkg.add("DataFrames")
Pkg.add("RDatasets")
using DataFrames
using RDatasets
```

How do we cope with missing data?

```
v = [x1, x2, x3, x4, x5]

mean(v)
```

```
v = [0.5, 0.6, 0.7, 0.8, 0.9]

mean(v)
```

```
v = [0.5, 0.6, 0.7, NA, 0.9]

mean(v)
```

The `NA` type:

- Represents a missing value
  - Like `NULL` in some systems
- Poisons other values
  - Like `NaN` for floating point numbers

```
1 + NA
1 > NA
isna(NA)
```

- `DataArray{T}` extends `Array{T}`
- `DataArray{T}` can store `T` or `NA`

```
dv = DataArray([1, 2, 3])
dv[1] = NA
join(dv, "::")
```

Convenience constructors:

- `datazeros()`
- `dataones()`
- `datafalses()`
- `datatrues()`
- `dataeye()`
- `datadiagm()`

```
dm = dataeye(4)
svd(dm)
```

Convenience converters:

- `dataint()`
- `datafloat()`
- `databool()`

```
dataint([1.0, 2.0, 3.0])
databool([1, 0, 1])
```

How do we store data efficiently?

`PooledDataArray{T}` compresses `DataArray{T}`

```
pda = PooledDataArray(["String 1",
                       "String 2",
                       "String 2",
                       "String 1"])
pda[1] = NA
```

```
levels(pda)
dump(pda)
```

How do we cope with heteregeneous data?

Name	Height	Weight	Gender
John Smith	73.0	NA	Male
Jane Doe	68.0	130	Female

```
df = DataFrame()
df["Name"] = DataVector["John Smith", "Jane Doe"]
df["Height"] = DataVector[73.0, 68.0]
df["Weight"] = DataVector[NA, 130]
df["Gender"] = DataVector["Male", "Female"]
df
```

- A `DataFrame` is a list of `DataVector`'s
- `DataFrame`'s allow mixed indexing:
  - Columns by number
  - Columns by name
  - Rows + Columns by number + number
  - Rows + Columns by number + name

```
df = DataFrame()
df["A"] = dataones(5)
df[2] = datazeros(Int, 5)
df
df[1, 1]
df[1, "A"]
```

```
movies = read_table(joinpath(julia_pkgdir(),
                             "DataFrames",
                             "test",
                             "data",
                             "movies.csv"))
```

```
head(movies)
tail(movies)
movies[1:2, 1:5]
```

```
iris = data("datasets", "iris")
colnames(iris)
coltypes(iris)
```

```
colnames!(iris, colnames(iris))
clean_colnames!(iris)
colnames(iris)
```

```
size(iris)

nrow(iris)
ncol(iris)
```

```
vcat(iris, iris)
hcat(iris, iris)

rbind(iris, iris)
cbind(iris, iris)
```

```
iris[1, 1] = NA
head(iris)
complete_cases(iris)
```

```
iris[complete_cases(iris), :]
complete_cases!(iris)
iris
```

```
cut([1, 2, 3, 4, 5, 6], [2, 3])
```

```
xtabs([1, 2, 3, 3, 4, 2, 3])
```

```
any(duplicated(iris))
any(duplicated(rbind(iris, iris)))
```

```
new_iris = rbind(iris, iris)
drop_duplicates!(new_iris)
new_iris
```

```
vector(iris["Species"])
vector(iris["Species"], Any)
```

```
matrix(iris)
matrix(iris[:, 2:3])
matrix(iris[:, 1:3])
matrix(iris[:, 1:3], Any)
```

```
with(iris, :(Petal_Length .* Petal_Width))
within!(iris,
        :(Petal_Area = Petal_Length .* Petal_Width))
head(iris)
```

Database operations on DataFrames:

- CRUD
- subset
- merge
- groupby

```
df1 = DataFrame({"a" => [1, 2, 3],
                 "b" => ["America", "Europe",
                         "Africa"]})
df2 = DataFrame({"a" => [1, 2, 4],
                 "c" => ["New World", "Old World",
                         "New World"]})
merge(df1, df2)
merge(df1, df2, "a", "inner")
merge(df1, df2, "a", "left")
merge(df1, df2, "a", "right")
merge(df1, df2, "a", "outer")
```

The Split-Apply-Combine Strategy:

- Segment data into groups
- Apply a function to each group independently
- Combine results into a single DataFrame

```
by(movies, "year", nrow)
subset(movies, :(year .== 1893))
```

```
by(movies, ["Action", "year"], nrow)
subset(movies, :(year .== 1893))
```

Let’s do some simple machine learning

```
using Clustering
iris = data("datasets", "iris")
k_means(matrix(iris[:, 2:5]), 3)
iris["Cluster"] =
  k_means(matrix(iris[:, 2:5]), 3).assignments
by(iris, ["Cluster", "Species"], nrow)
```

```
using kNN, Resampling
iris = data("datasets", "iris")
train, test = splitrandom(iris, 0.80)
test["Guess"] = knn(matrix(train[:, 2:5]),
                    matrix(test[:, 2:5]),
                    vector(train[:, 6]),
                    10)
by(test, ["Species", "Guess"], nrow)
mean(test["Guess"] .== test["Species"])
```

Simple exercises:

- Create some DataArray's
  - A DataArray containing the first five primes
  - A DataArray containing only NA's
  - A DataArray containing DataArray's
- Create some DataFrame's
  - A 4x3 DataFrame w/:
    - 4 String's
    - 4 Int's
    - 4 ComplexPair's

Advanced exercises:

- For each species in `iris`:
    - Find the median petal area
    - Find the variance of the petal area
    - Find the centroid of the sepal and petal dimensions