Introduction to sparklyr

Contents

0.1 Preliminiaries . . . 1
0.2 Connecting . . . 1
0.3 Data input/output . . . 1
0.4 Data wrangling . . . 2

1 Calculate the mean of each column **2**
1.1 Plots . . . 3
1.2 Models, in brief . . . 3
1.3 Logistic regression . . . 4
1.4 Disconnecting . . . 6

We will largely follow chapters **2** and **3** of **Mastering Spark with R**, https://therinspark.com.

First install the following packages if you do not already have them already, and load them with the `library()` function:

```
library(sparklyr)
library(dplyr)
library(ggplot2)
library(knitr)
```

0.1 Preliminiaries

If you are working on EIDF, first make sure that the default working directory in RStudio is your folder. In RStudio, select Tools -> Global Options. Change the default working directory to be /work/eidf071/eidf071/.

To confirm the change has taken effect, close and then reopen RStudio, and type getpw() into the console. It should show your working directory correctly as above.

0.2 Connecting

```
sc = spark_connect(master = 'local')
```

The following code will take the built-in mtcars dataset, stored in an R dataframe, and put it into a spark dataframe. We will use this dataset in many of our examples.

```
cars = copy_to(sc, mtcars, overwrite = TRUE)
```

0.3 Data input/output

0.3.1 Write to a csv file

This will create a folder in your working directory called `cars.csv`. It contains a csv with the cars data in it.

```
{ block4} spark_write_csv(cars, "cars.csv")
```

Note that running this more than once will result in an error because spark_write_csv will not overwrite a folder which is already created. You may need to delete the folder before running the code again.

0.3.2 Read from a csv file

```
spark_read_csv(sc, 'cars.csv') %>%
  head() %>%
  kable()
```

mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

0.4 Data wrangling

Familiar commands from dplyr work as you would expect, but now they instead connect to Spark and would be run in parallel across the cluster.

0.4.1 Create a new column

```
cars = mutate(cars, transmission = ifelse(am == 0, 'automatic', 'manual'))
```

0.4.2 Select columns

```
select(cars, am, transmission) %>%
  head() %>%
  kable()
```

am	transmission
1	manual
1	manual
1	manual
0	automatic
0	automatic
0	automatic

1 Calculate the mean of each column

```
summarise_all(cars, mean, na.rm = TRUE) %>%
  kable()
```

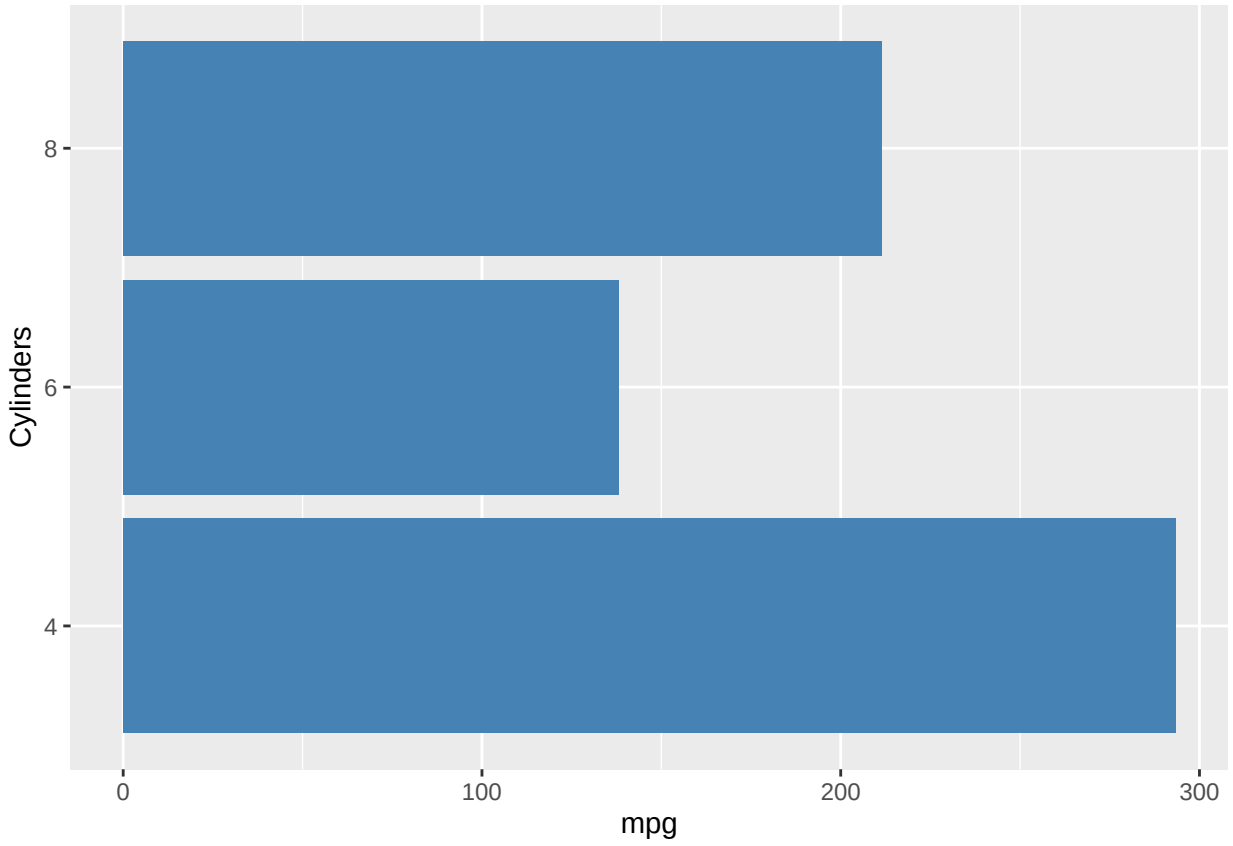
mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	transmission
20.09062	6.1875	230.7219	146.6875	3.596563	3.21725	17.84875	0.4375	0.40625	3.6875	2.8125	NA

1.1 Plots

Creating a plot isn't usually highly computationally demanding. Therefore, sparklyr does not have a full-fledged equivalent of ggplot. It is typically best to perform all data manipulations in Spark, then bring the result back to R using the collect() command. Finally, we use the regular ggplot package to make the graph.

```
# Data manipulations are done first using spark
car_group = cars %>%
  group_by(cyl) %>%
  summarise(mpg = sum(mpg, na.rm = TRUE)) %>%
  # `collect` brings the spark dataframe back to a regular R dataframe
  collect()

# Now use ggplot on the R dataframe car_group
ggplot(aes(as.factor(cyl), mpg), data = car_group) +
  geom_col(fill = 'SteelBlue') +
  xlab('Cylinders') +
  coord_flip()
```



1.2 Models, in brief

We will go into more details about these models in coming lectures.

1.2.1 OLS

```
ols_model = ml_linear_regression(cars, mpg ~ hp + disp)
summary(ols_model)
```

```
## Deviance Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7945 -2.3036 -0.8246  1.8582  6.9363
##
## Coefficients:
## (Intercept)          hp        disp
## 30.73590425 -0.02484008 -0.03034628
##
## R-Squared: 0.7482
## Root Mean Squared Error: 2.976
```

1.3 Logistic regression

The command `ml_logistic_regression` can be used to train a multinomial model, where the dependent variable has more than two categories. However, it does not report standard deviations of parameter estimates.

```
lr_model = ml_logistic_regression(cars, am ~ hp + disp)
summary(lr_model)
```

```
## Coefficients:
## (Intercept)          hp        disp
##  1.40342475  0.12170163 -0.09517967
```

The command `ml_generalized_linear_regression` can also be used to train a logistic model with binary variable, but **dependent variables with more than two categories are not supported!** However, it does report standard deviations of parameter estimates.

```
lr_model = ml_generalized_linear_regression(cars, am ~ hp + disp, family = 'binomial')
summary(lr_model)
```

```
## Deviance Residuals:
##       Min        1Q    Median        3Q       Max
## -1.966529 -0.309001 -0.001701  0.393378  1.368229
##
## Coefficients:
## (Intercept)          hp        disp
##  1.40342203  0.12170173 -0.09517972
##
## (Dispersion parameter for binomial family taken to be 1)
##
##    Null  deviance: 43.2297 on 31 degrees of freedom
## Residual deviance: 16.7129 on 29 degrees of freedom
## AIC: 22.713
```

1.3.1 Multilayer perceptron

```
mlp_model = ml_multilayer_perceptron_classifier(
  cars,
  am ~ hp + disp,
  layers = c(2, 8, 8, 2)
)
predictions = ml_predict(mlp_model, cars)

select(predictions, prediction, probability_0, probability_1) %>%
```

```
  head() %>%
  kable()
```

prediction	probability_0	probability_1
0	0.8571260	0.1428740
0	0.8571260	0.1428740
1	0.0909109	0.9090891
0	0.8571260	0.1428740
0	0.8571260	0.1428740
0	0.8571260	0.1428740

Gradient boosted tress

Classification trees:

```
gbt_model = ml_gradient_boosted_trees(cars, am ~ hp + disp, type = 'classification')
predictions = ml_predict(gbt_model, cars)

select(predictions, prediction, probability_0, probability_1) %>%
  head() %>%
  kable()
```

prediction	probability_0	probability_1
1	0.0436465	0.9563535
1	0.0436465	0.9563535
1	0.0436465	0.9563535
0	0.9563535	0.0436465
0	0.9563535	0.0436465
0	0.9563535	0.0436465

Regression trees:

```
gbt_model = ml_gradient_boosted_trees(cars, mpg ~ hp + disp, type = 'regression')
predictions = ml_predict(gbt_model, cars)

select(predictions, mpg, prediction) %>%
  head() %>%
  kable()
```

mpg	prediction
21.0	21.00351
21.0	21.00351
22.8	22.80000
21.4	21.39966
18.7	18.69963
18.1	18.10059

1.3.2 Other models

Apache Spark supports many other models - I have just chosen a few to look at more closely. I encourage you to explorer others! See documentation here: https://spark.apache.org/docs/latest/ml-classification-regression.html

1.4 Disconnecting

The following code chukn disconnects from the cluster. You should always do this after your job has been run.

```
spark_disconnect(sc)
```