Data science and analysis in Neuroscience

Kevin Allen

November 21, 2019

Today's plan

1. Review exercises
2. RStudio editor and .R files
3. git repositories, part 1
4. ggplot2
5. Get behavioral data into R
6. dplyr
7. Exercises: dplyr and ggplot2

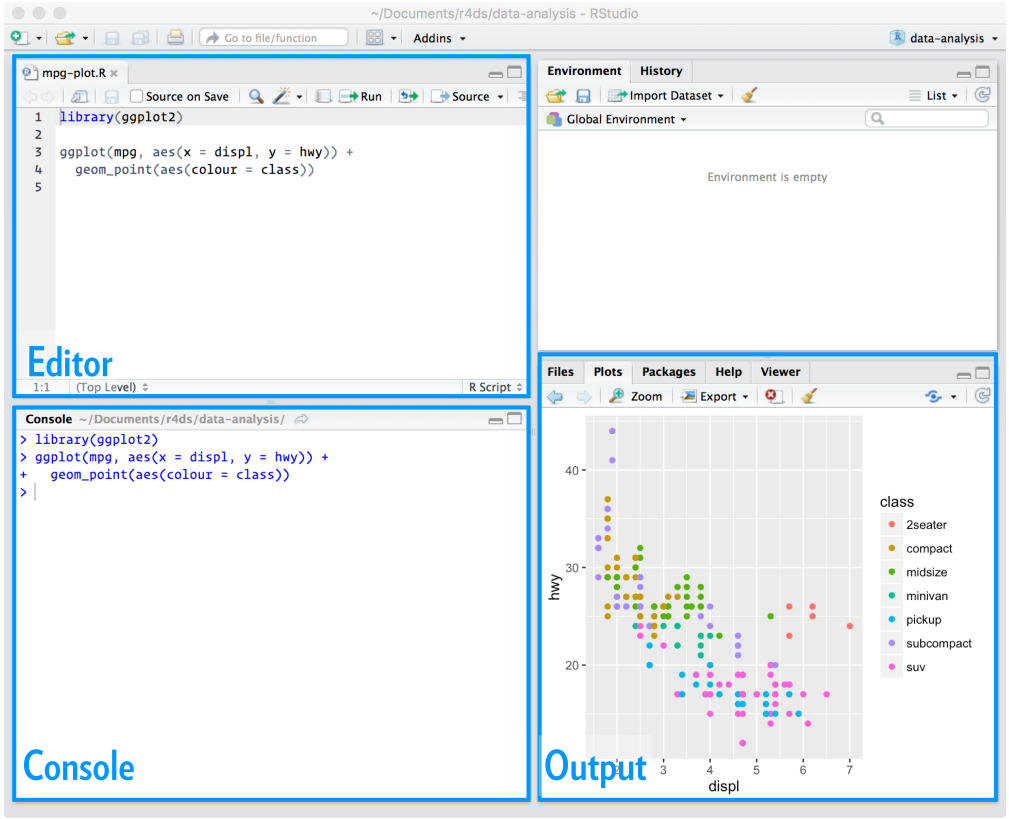
Reactivation exercises

1. What is the difference between `x = 5` and `x == 5`?
2. `y = "moon"`, what type of object is `y`? What are the other types?
3. How can I find out what type an object is?
4. What is a data frame?
5. How do I get information regarding the use of a built-in function (e.g., `mean`)?

RStudio editor

A place to store the code you care about.

Save code into a .R file for later use or sharing.



RStudio editor

1. Open a new R script (File/New File/R script).
2. Execute a single expression: Cmd/Ctrl-Enter
3. Execute the entire script: Cmd/Ctrl-Shift-S

RStudio editor

Create an R script. Write this code in the RStudio editor, run the code, and save it as `myFirstPlot.R`.

```
library(tidyverse)
ggplot(data=mpg) +
  geom_point(mapping=aes(x=displ,y=hwy))
```

git repositories

1. Version control system to track changes in files in a directory
2. Works with any programming languages.
3. Takes snapshot of your files at any time
4. Supports cooperative work
5. Local and remote repositories (e.g., GitHub servers)
6. Revert back to previous version

git repositories

Why should I even bother?

1. Prevent catastrophes (a real example)
2. Force you to keep your code in order
3. Easy way to share your code along with publications.

I recommend creating a repository for any projects that involve programming.

git repositories

An example of a repository: The course repository on GitHub.

https://github.com/kevin-allen/dataNeuroCourse

Live example.

We will come back to git repositories when you have more code!

An introduction to ggplot2

1. `ggplot2` is an R package for visualization.
2. It is relatively easy to learn.
3. You get nice looking graphs relatively quickly.
4. It works well with data frames.
5. R also has base graphics, `plot()`

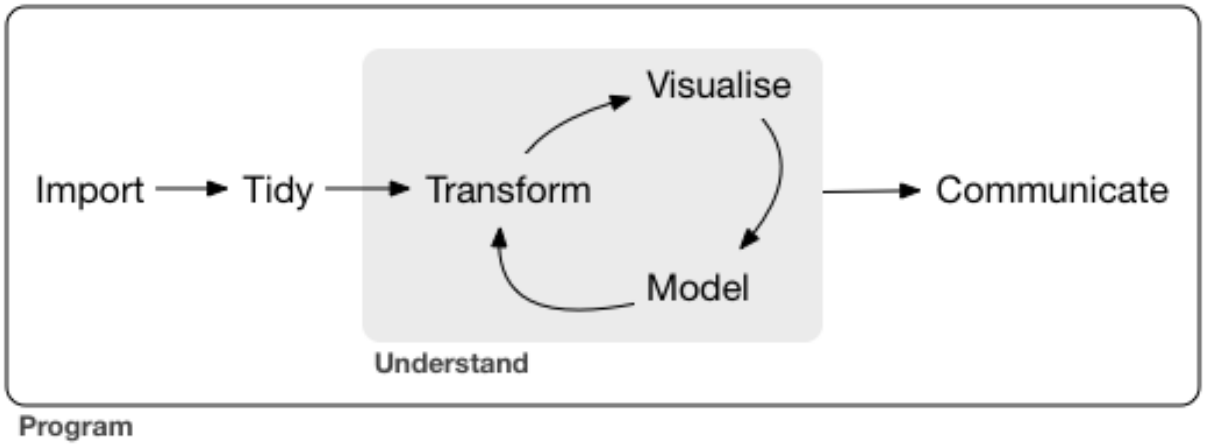


Figure from "R for Data Science"

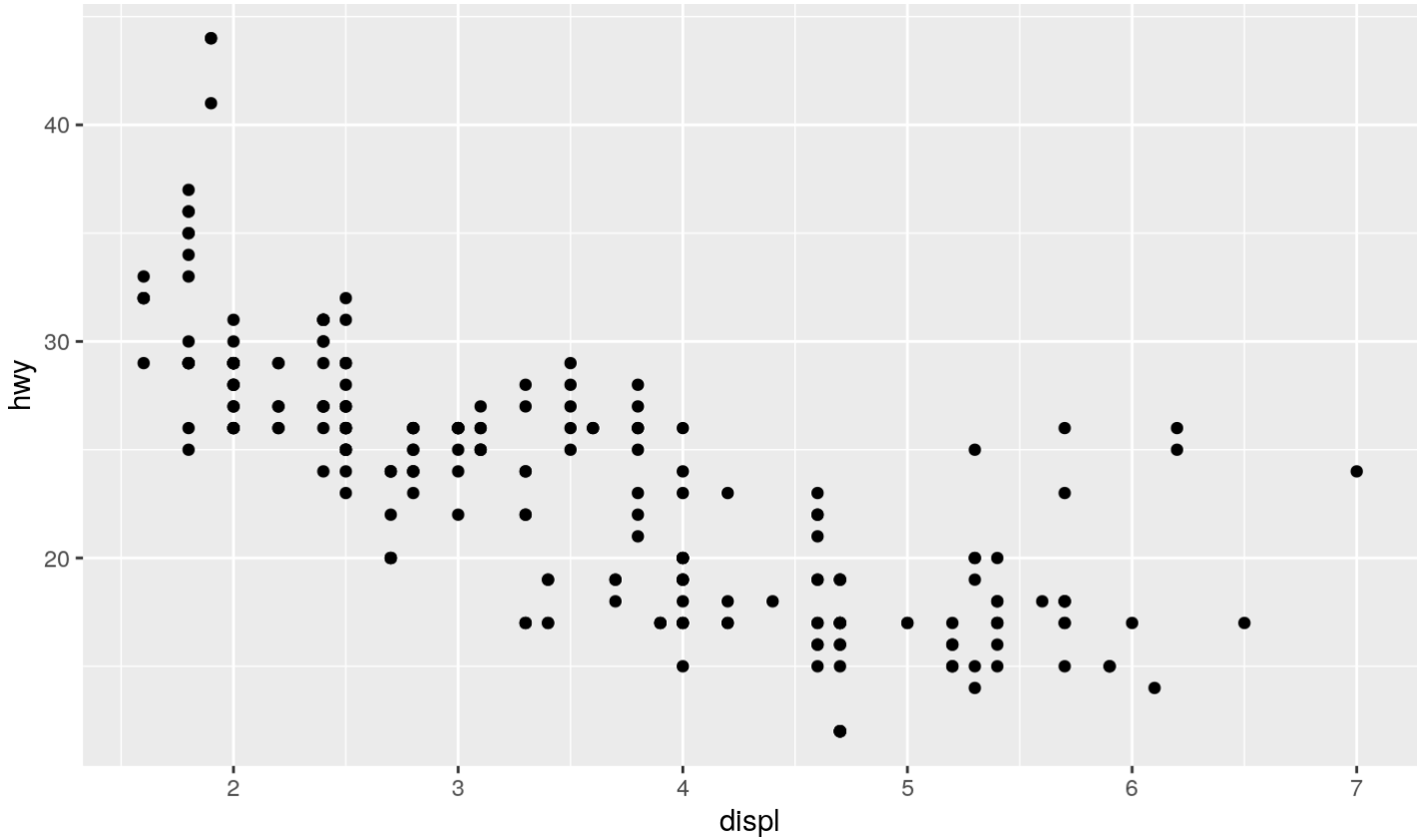
Load the car dataset (mpg)

```
library(tidyverse)
print(mpg)
```

```
## # A tibble: 234 x 11
##    manufacturer model displ  year   cyl trans drv     cty   hwy fl    class
##    <chr>        <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
##  1 audi         a4      1.8  1999     4 auto… f        18    29 p     comp…
##  2 audi         a4      1.8  1999     4 manu… f        21    29 p     comp…
##  3 audi         a4      2    2008     4 manu… f        20    31 p     comp…
##  4 audi         a4      2    2008     4 auto… f        21    30 p     comp…
##  5 audi         a4      2.8  1999     6 auto… f        16    26 p     comp…
##  6 audi         a4      2.8  1999     6 manu… f        18    26 p     comp…
##  7 audi         a4      3.1  2008     6 auto… f        18    27 p     comp…
##  8 audi         a4 q…   1.8  1999     4 manu… 4        18    26 p     comp…
##  9 audi         a4 q…   1.8  1999     4 auto… 4        16    25 p     comp…
## 10 audi         a4 q…   2    2008     4 manu… 4        20    28 p     comp…
## # … with 224 more rows
```

Create a plot

```
ggplot(data=mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



Template

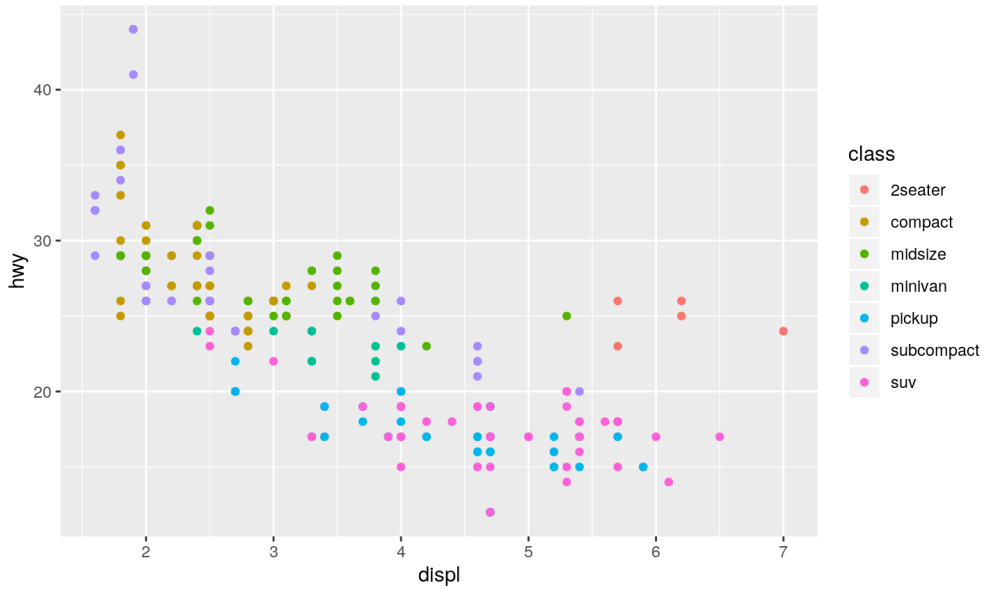
```
ggplot(data=<DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

1. `ggplot` creates an empty graph and coordinate system for the data
2. `geom` functions add layers to the plot
3. `geom` functions need a mapping argument paired with `aes()`

Aesthetic Mapping

Aesthetics include the x-axis and y-axis, together with size, shape, color of the points.

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```



Shape and color work better with categorical variables.

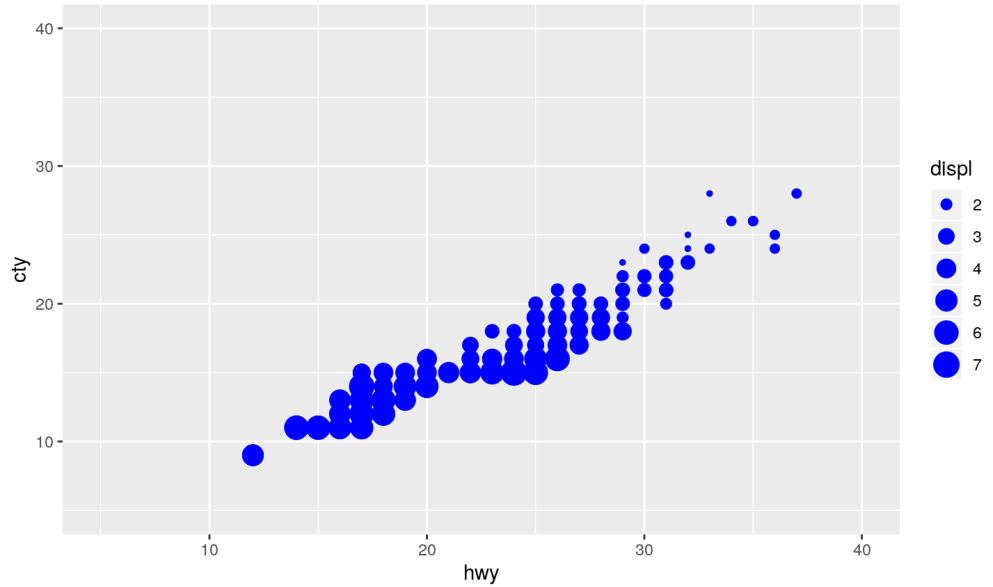
Exercise with ggplot2

1. What plot could you produce to investigate whether cars are more efficient on the highway or in a city?
2. How could you add information about the displacement of the motor in your graph?

Solution

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = hwy, y = cty, size=displ),
             color = "blue")+
  xlim(5,40) +
  ylim(5,40)
```

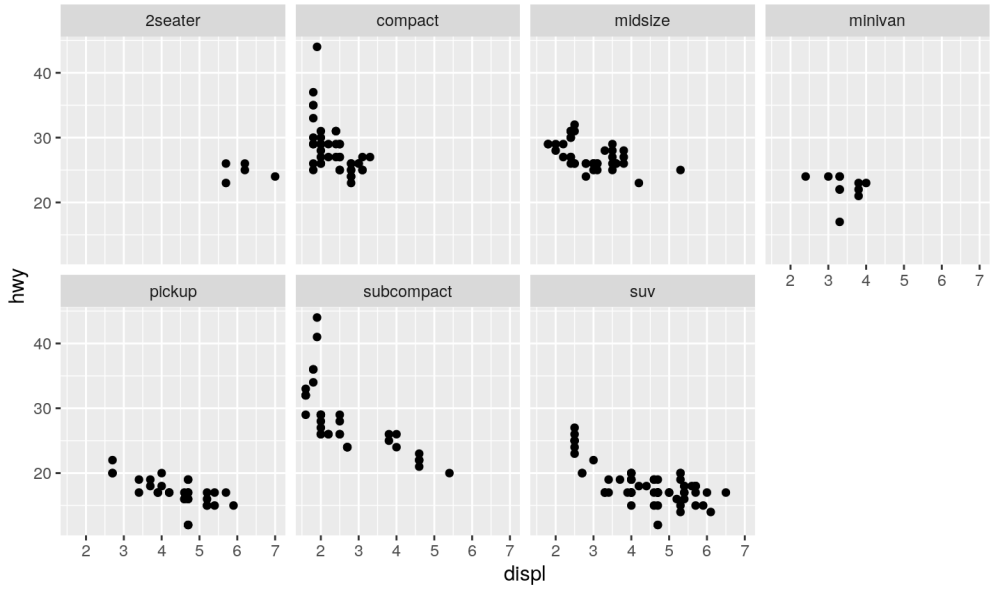
```
## Warning: Removed 3 rows containing missing values (geom_point).
```



Facets

Split your plot into facets

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~class, nrow = 2)
```



Geometric objects

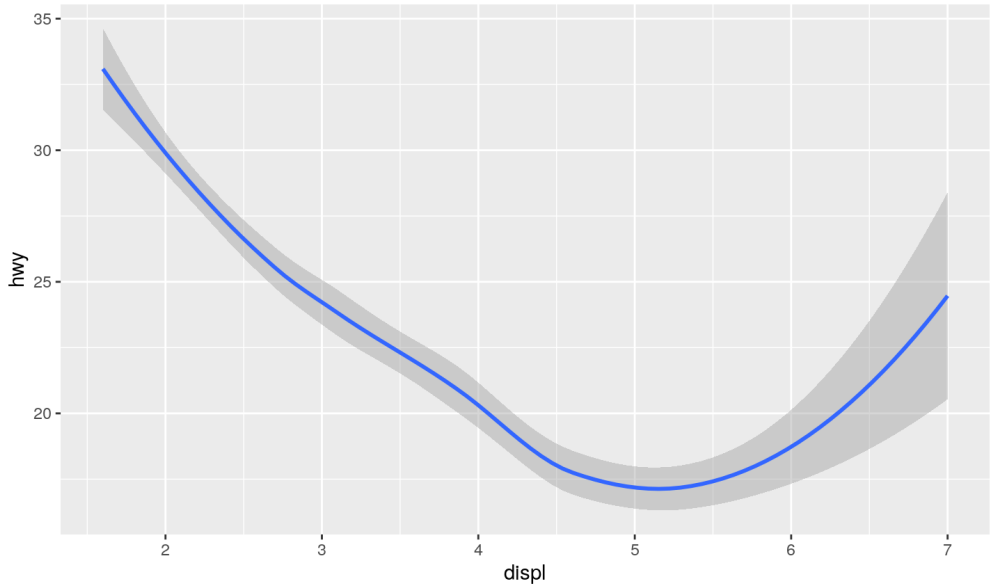
- Used to show different types of graphs
- There are up to 30 types of geoms.
- You can mix geoms.

Geometric objects

geom_smooth

```
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

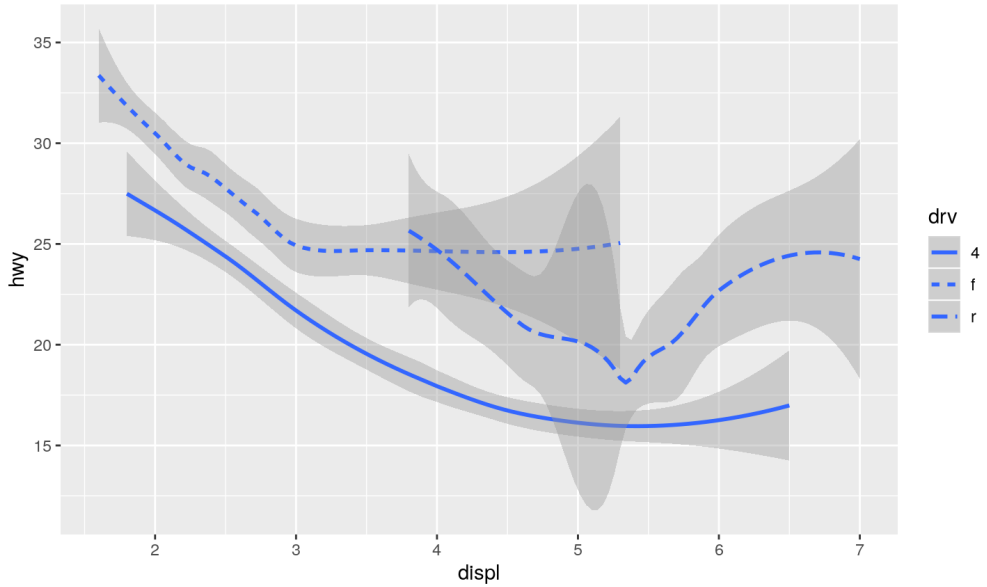


Geometric objects

geom_smooth

```
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy, linetype = drv))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

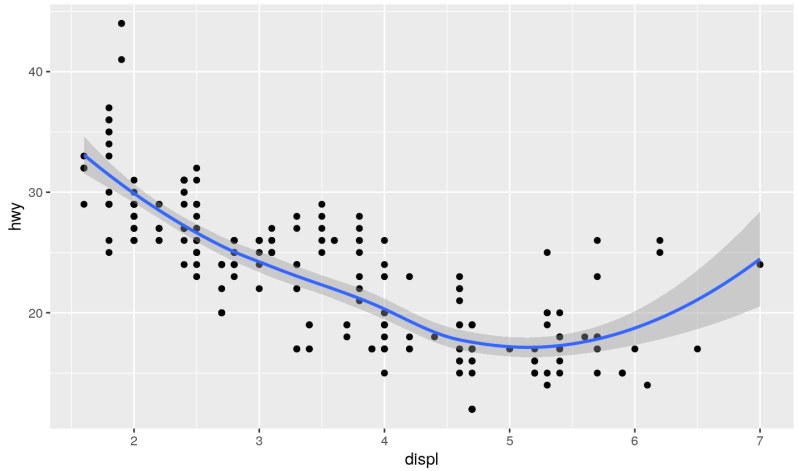


Geometric object

Combining several geoms

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth()
```

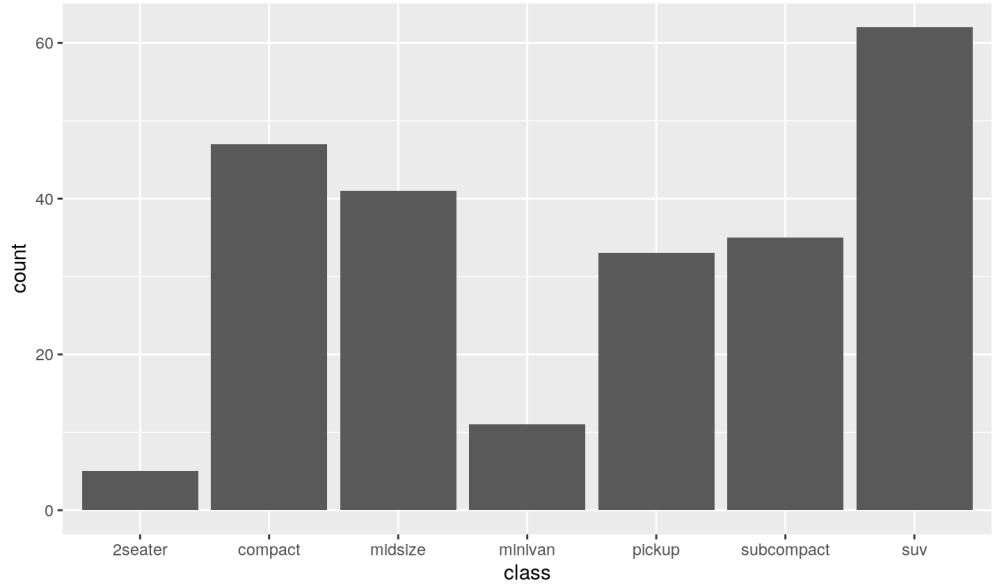
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Count and distributions

Categorical variable

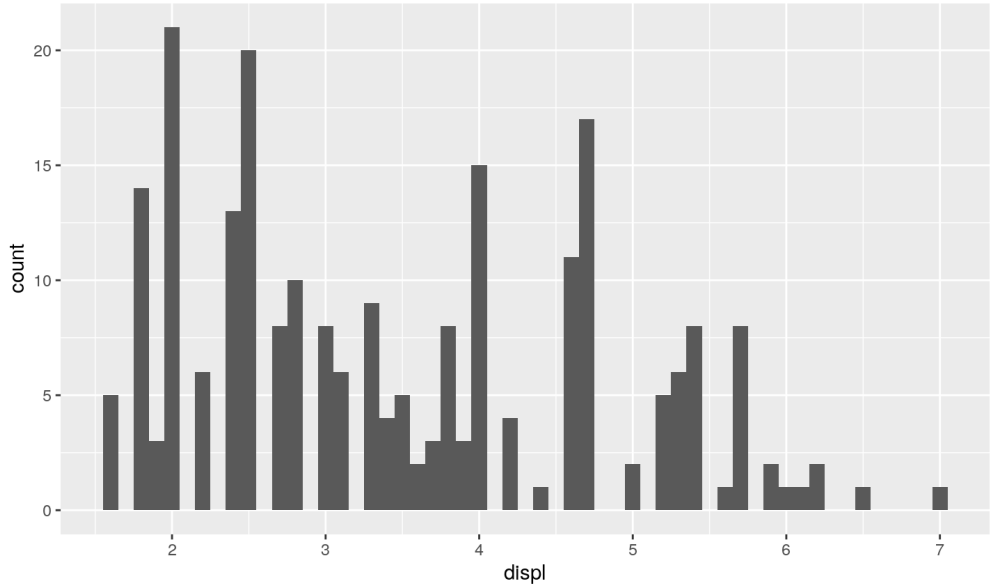
```
ggplot(data = mpg) +
  geom_bar(mapping = aes(x=class))
```



Count and distribution

Continuous variable

```
ggplot(data = mpg) +
  geom_histogram(mapping = aes(x=displ),binwidth = 0.1)
```

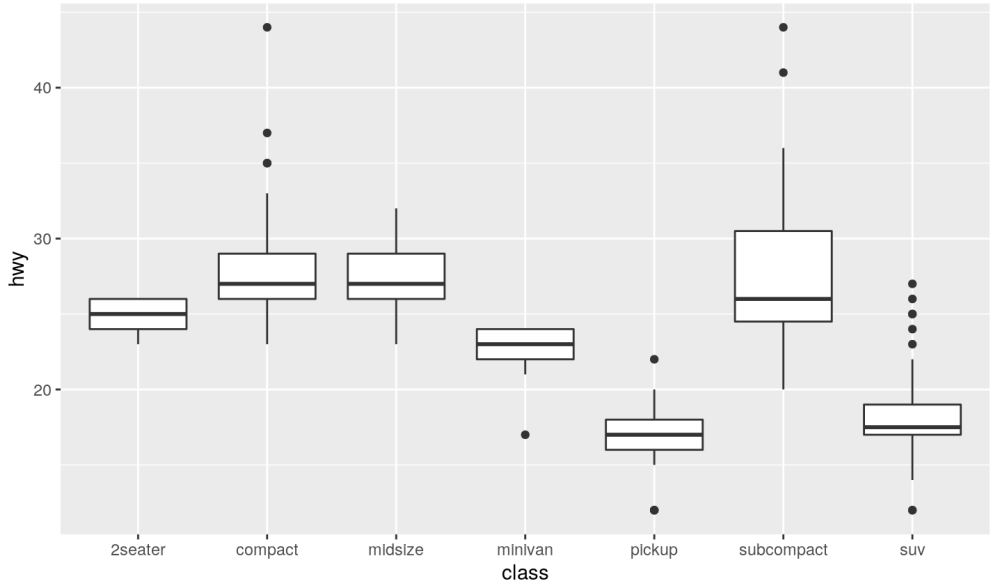


Exercise

1. Produce a boxplot showing the highway consumption for the different classes of cars.

Solution

```
ggplot(data = mpg) +
  geom_boxplot(mapping = aes(x=class,y=hwy))
```



Behavioural data set

1. We trained mice on a spatial memory task called a rewarded alternation task.
2. The t-maze has a start arm and two goal arms (right and left)
3. Each trial has a forced run and choice run.
4. During the forced run, the animal is forced to go to one goal arm and receive a reward.
5. During the choice run, the animal can choose which goal arm to enter. Only the non-visited arm contains food.

Download a data set

Download the data set: https://tinyurl.com/r52pxm5 or here: https://github.com/kevin-allen/dataNeuroCourse/tree/master/dataSets

Load the data set in R

```
# path to your downloaded file
myFile="~/repo/dataNeuroCourse/dataSets/tmaze.csv"

df<-read_csv(myFile)


## Parsed with column specification:
## cols(
##   mouse = col_character(),
##   date = col_date(format = ""),
##   injection = col_character(),
##   block = col_double(),
##   trialNo = col_double(),
##   sample = col_character(),
##   choice = col_character()
## )
```

Inspect the data set

```
print(df)
```

```
## # A tibble: 1,120 x 7
##    mouse  date       injection block trialNo sample choice
##    <chr>  <date>     <chr>     <dbl>   <dbl> <chr>  <chr>
##  1 Mn4656 2019-10-09 Saline        1       1 L      R
##  2 Mn4656 2019-10-09 Saline        1       2 L      L
##  3 Mn4656 2019-10-09 Saline        1       3 R      R
##  4 Mn4656 2019-10-09 Saline        1       4 L      R
##  5 Mn4656 2019-10-09 Saline        1       5 R      L
##  6 Mn4656 2019-10-09 Saline        1       6 R      R
##  7 Mn4656 2019-10-09 Saline        1       7 L      R
##  8 Mn4656 2019-10-09 Saline        1       8 L      R
##  9 Mn4656 2019-10-09 Saline        1       9 R      L
## 10 Mn4656 2019-10-09 Saline        1      10 L      R
## # … with 1,110 more rows
```

```
view(df)
```

tidyverse

`tidyverse` is a collection of R packages for data science.

Advantages of tidyverse over built-in core R functions

1. Code is easier to understand
2. Code is easier to read
3. Easier to learn

Disadvantage

1. You need tidy data frames (like our t-maze data)

tidyverse

```
library(tidyverse)
```

Data transformation with dplyr

1. Pick observations (rows) by their values: `filter()`
2. Reorder the rows: `arrange()`
3. Pick variable (columns) by names: `select()`
4. Create new variables from existing variable: `mutate()`
5. Collapse many values down to a single summary: `summarize()`

filter()

Pick observations (rows) by their values. It uses logicals (TRUE or FALSE) to select the rows.

```
filter(df, injection=="Saline")
filter(df, mouse=="Mn4656",block==1) # this uses AND, both needs to be true
filter(df, block %in% c(1,2,3))
```

Main idea:

mouse <chr>	date <date>	injection <chr>	block <dbl>	trialNo <dbl>	sample <chr>	choice <chr>
Mn4656	2019-10-09	Saline	1	1	L	R
Mn4656	2019-10-09	Saline	1	2	L	L
Mn4656	2019-10-09	Saline	1	3	R	R
Mn4656	2019-10-09	Saline	1	4	L	R
Mn4656	2019-10-09	Saline	1	5	R	L
Mn4656	2019-10-09	Saline	1	6	R	R
Mn4656	2019-10-09	Saline	1	7	L	R
Mn4656	2019-10-09	Saline	1	8	L	R
Mn4656	2019-10-09	Saline	1	9	R	L
Mn4656	2019-10-09	Saline	1	10	L	R

1-10 of 1,120 rows Previous 1 2 3 4 5 6 … 100 Next

arrange()

Change the order of rows

```
arrange(df,trialNo)
arrange(df,desc(trialNo))
```

select()

It drops all columns that are not mentioned.

```
select(df,mouse,sample,choice)
select(df,starts_with("bl"))
select(df,ends_with("e"))
```

Main idea:

mouse <chr>	date <date>	injection <chr>	block <dbl>	trialNo <dbl>	sample <chr>	choice <chr>
Mn4656	2019-10-09	Saline	1	1	L	R
Mn4656	2019-10-09	Saline	1	2	L	L
Mn4656	2019-10-09	Saline	1	3	R	R
Mn4656	2019-10-09	Saline	1	4	L	R
Mn4656	2019-10-09	Saline	1	5	R	L
Mn4656	2019-10-09	Saline	1	6	R	R
Mn4656	2019-10-09	Saline	1	7	L	R
Mn4656	2019-10-09	Saline	1	8	L	R
Mn4656	2019-10-09	Saline	1	9	R	L
Mn4656	2019-10-09	Saline	1	10	L	R

1-10 of 1,120 rows Previous 1 2 3 4 5 6 ... 100 Next

mutate()

Add a new variable from old ones.

```
mutate(df,correct=sample!=choice)

# This was actually useful! Save the results
df<-mutate(df,correct=sample!=choice)
```

mutate

Check if we got what we wanted

```
select(df,sample,choice,correct)
```

```
## # A tibble: 1,120 x 3
##    sample choice correct
##    <chr>  <chr>  <lgl>
##  1 L      R      TRUE
##  2 L      L      FALSE
##  3 R      R      FALSE
##  4 L      R      TRUE
##  5 R      L      TRUE
##  6 R      R      FALSE
##  7 L      R      TRUE
##  8 L      R      TRUE
##  9 R      L      TRUE
## 10 L      R      TRUE
## # … with 1,110 more rows
```

summarize()

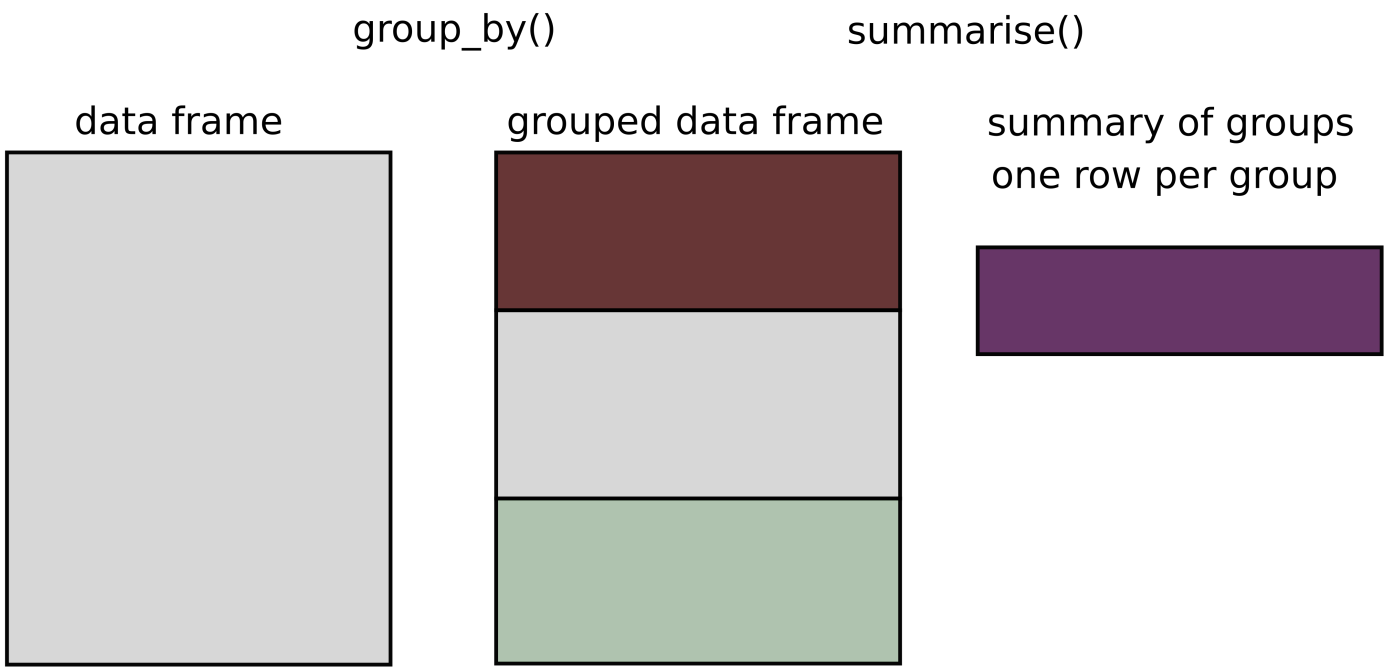
Collapse several rows into a single row.

```
summarize(df, performance = mean(correct))
```

```
## # A tibble: 1 x 1
##   performance
##         <dbl>
## 1       0.716
```

group_by and summarise()

Collapse according to some groups.



group_by and summarise()

Group by mouse, then get the mean performance.

```
by_df<-group_by(df,mouse)
summarise(by_df,performance= mean(correct))
```

```
## # A tibble: 7 x 2
##   mouse  performance
##   <chr>        <dbl>
## 1 Mn4656       0.706
## 2 Mn4672       0.712
## 3 Mn4673       0.725
## 4 Mn7712       0.794
## 5 Mn7735       0.65
## 6 Mn829        0.744
## 7 Mn848        0.681
```

group_by and summarise()

Group by mouse and block, then count the number of trials.

```
by_df<-group_by(df,mouse,block)
summarise(by_df,noTrials= n())
```

```
## # A tibble: 91 x 3
## # Groups:   mouse [7]
##    mouse  block noTrials
##    <chr>  <dbl>    <int>
##  1 Mn4656     1       20
##  2 Mn4656     2       20
##  3 Mn4656     3       20
##  4 Mn4656     4       10
##  5 Mn4656     5       10
##  6 Mn4656     6       10
##  7 Mn4656     7       10
##  8 Mn4656     8       10
##  9 Mn4656     9       10
```

group_by and summarise()

Group by mouse and block, count trials and get the mean performance.

```
by_df<-group_by(df,mouse,block)
summarise(by_df,noTrials = n(), performance= mean(correct))
```

```
## # A tibble: 91 x 4
## # Groups:   mouse [7]
##    mouse  block noTrials performance
##    <chr>  <dbl>    <int>       <dbl>
##  1 Mn4656     1       20        0.85
##  2 Mn4656     2       20        0.75
##  3 Mn4656     3       20        0.55
##  4 Mn4656     4       10        0.7
##  5 Mn4656     5       10        0.5
##  6 Mn4656     6       10        0.7
##  7 Mn4656     7       10        1
##  8 Mn4656     8       10        0.5
##  9 Mn4656     9       10        0.7
```

Using Pipe

Combine multiple operations with the Pipe: %>%

%>% is like "then"

```
do_thid %>% do_that %>% do_something_else
```

Using Pipe

```
## without Pipe
df_mouse<- filter(df,mouse=="Mn4656")
df_mouse_sel<- select(df_mouse,mouse,block,trialNo,correct)
df_mouse_sel_gb<-group_by(df_mouse_sel,block)
summarise(df_mouse_sel_gb,performance=mean(correct))

## with Pipe
df %>%
  filter(mouse=="Mn4656") %>%
  select(mouse,block,trialNo,correct) %>%
  group_by(block) %>%
  summarize(performance=mean(correct))
```

Putting it all together (dplyr and ggplot)

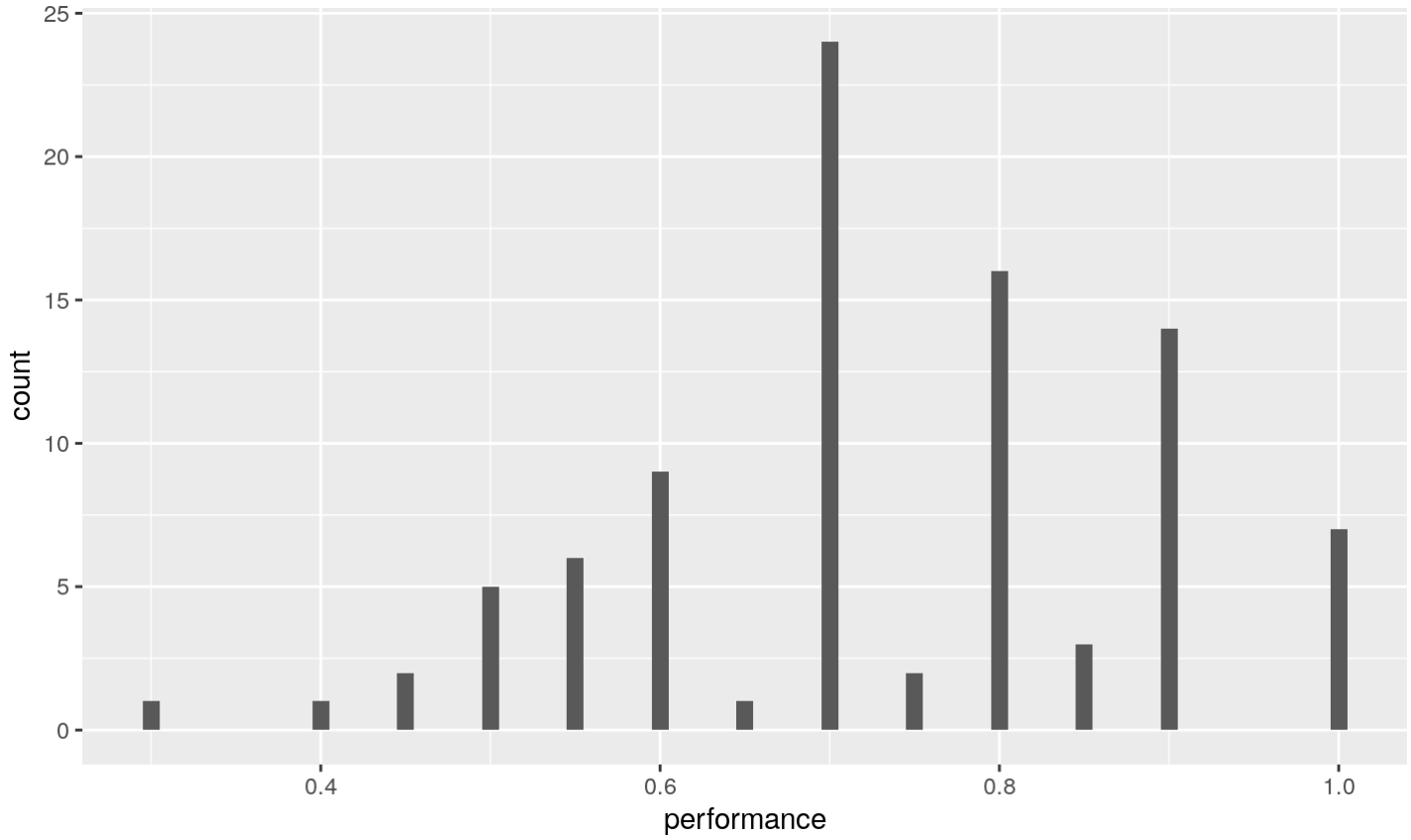
- Create a data frame in which you have the performance of single mice on single blocks. Use the Pipe!

```
new_df<-df %>%
  group_by(mouse,block) %>%
  summarise(performance = mean(correct))
```

Putting it all together (dplyr and ggplot)

- Plot a histogram showing the distribution of performance on every block and mouse (mouse1-block1, mouse1-block2, mouseN-blockN).

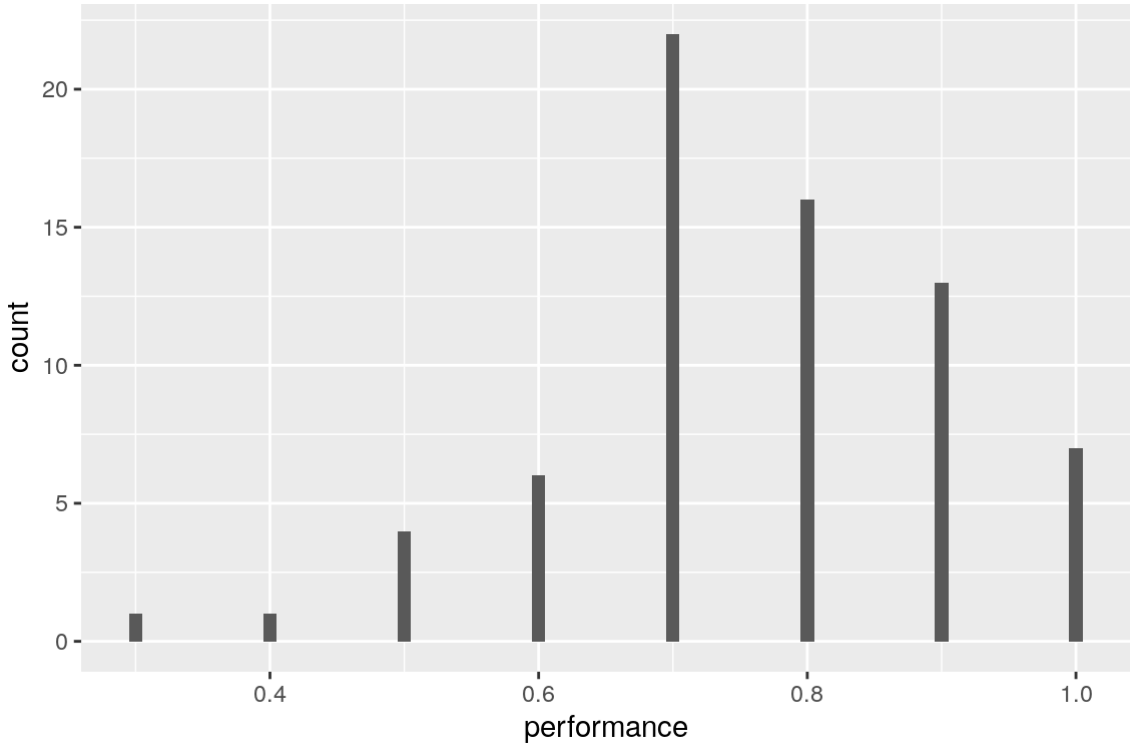
```
new_df<-df %>%
  group_by(mouse,block) %>%
  summarise(performance = mean(correct))
ggplot (data=new_df) +
  geom_histogram(mapping = aes(x = performance),binwidth=0.01)
```



Put it all together (dplyr and ggplot)

- The first 3 days of training (e.g, blocks) had 20 trials instead of 10. Plot the distribution of performance on every block and mouse, but only for blocks with 10 trials.

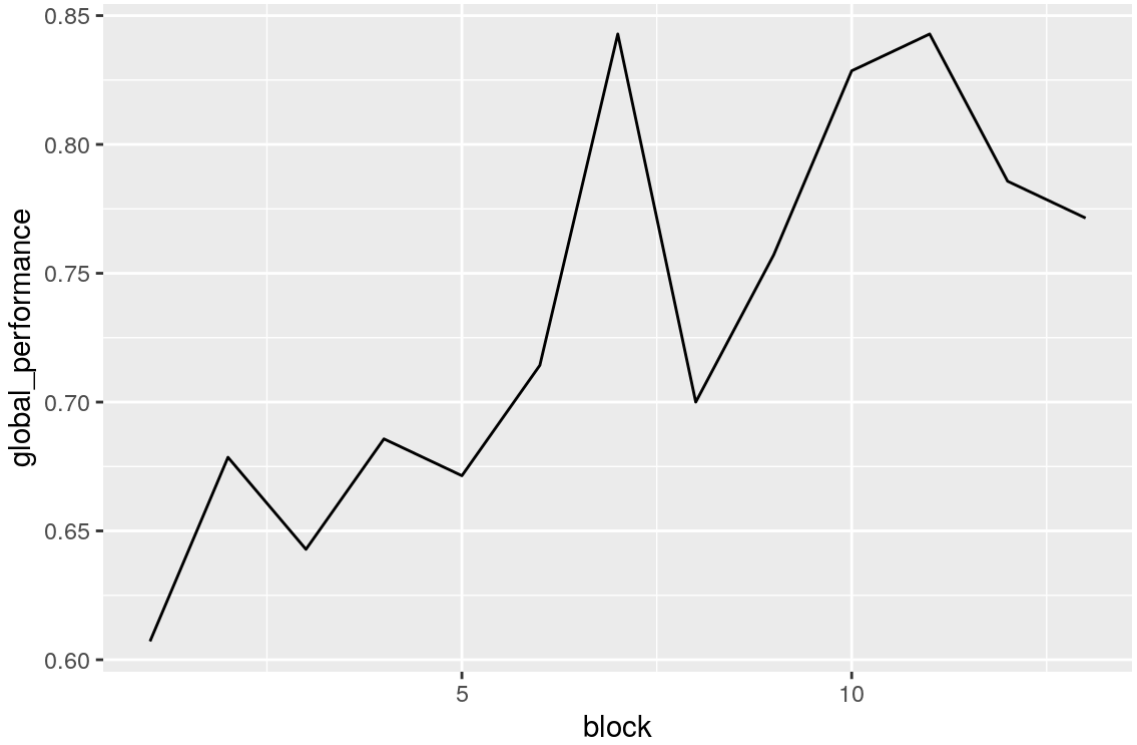
```
new_df<- df %>%
  group_by(mouse,block) %>%
  summarise(nTrials = n(), performance = mean(correct)) %>%
  filter(nTrials==10)
ggplot (data=new_df) +
  geom_histogram(mapping = aes(x = performance),binwidth=0.01)
```



Put it all together (dplyr and ggplot)

- Make a plot showing the mean performance of all mice on each block. Use geom_line or geom_point.

```
df %>% group_by(mouse,block) %>%
  summarise(performance = mean(correct)) %>%
  group_by(block) %>%
  summarise(global_performance = mean(performance)) %>%
  ggplot()+
  geom_line(mapping=aes(x=block,y=global_performance))
```

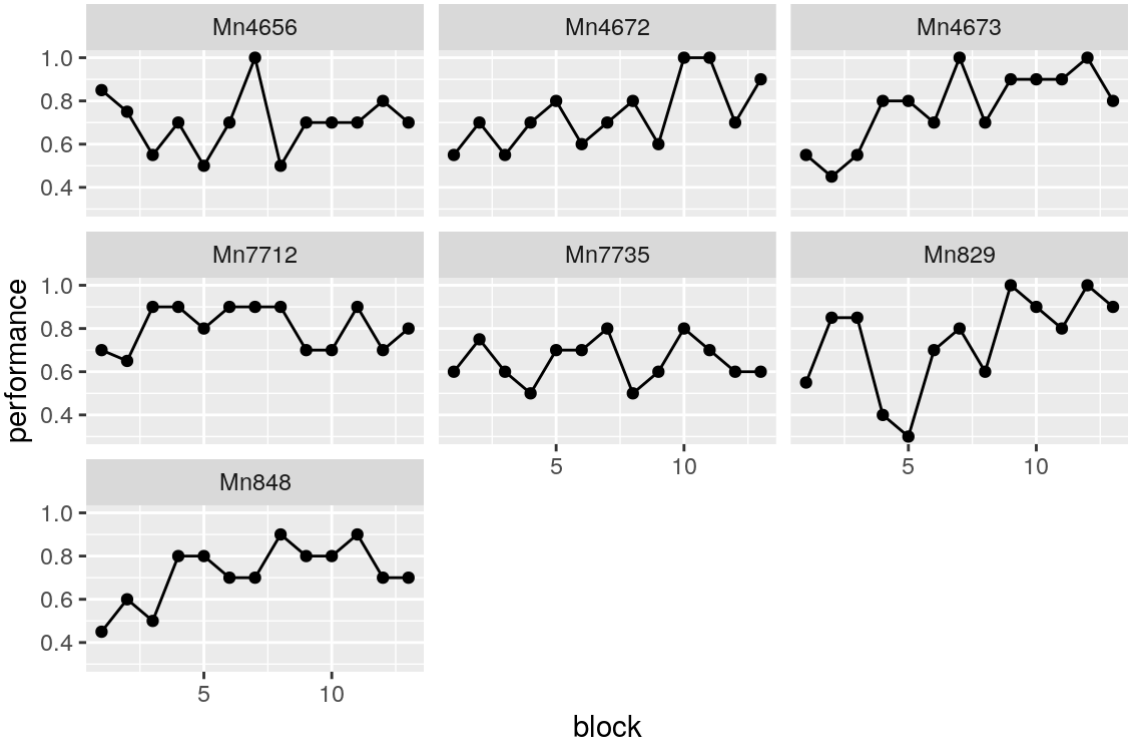


Put it all together (dplyr and ggplot)

- Remember the facet_wrap function in ggplot? It can be used to plot several graphs into the same plot

Use facet_wrap to display the performance of each mouse separately

```
df %>% group_by(mouse,block) %>%
  summarise(performance = mean(correct)) %>%
  ggplot()+
  geom_point(mapping=aes(x=block,y=performance))+
  geom_line(mapping=aes(x=block,y=performance))+
  facet_wrap(~mouse)
```



Next lecture

1. More exercises on dplyr and ggplot with a different data frame
2. Combining information from 2 data frames
3. Create your own git repository on github
4. Saving your data to a file
5. Dealing with NAs

For the next lecture

1. Review ggplot and dplyr exercises.
2. Install git on your computer and clone the course repository (https://github.com/kevin-allen/dataNeuroCourse)
3. Read chapters 1, 2 and 3 of R for Data Science