Data science and analysis in Neuroscience

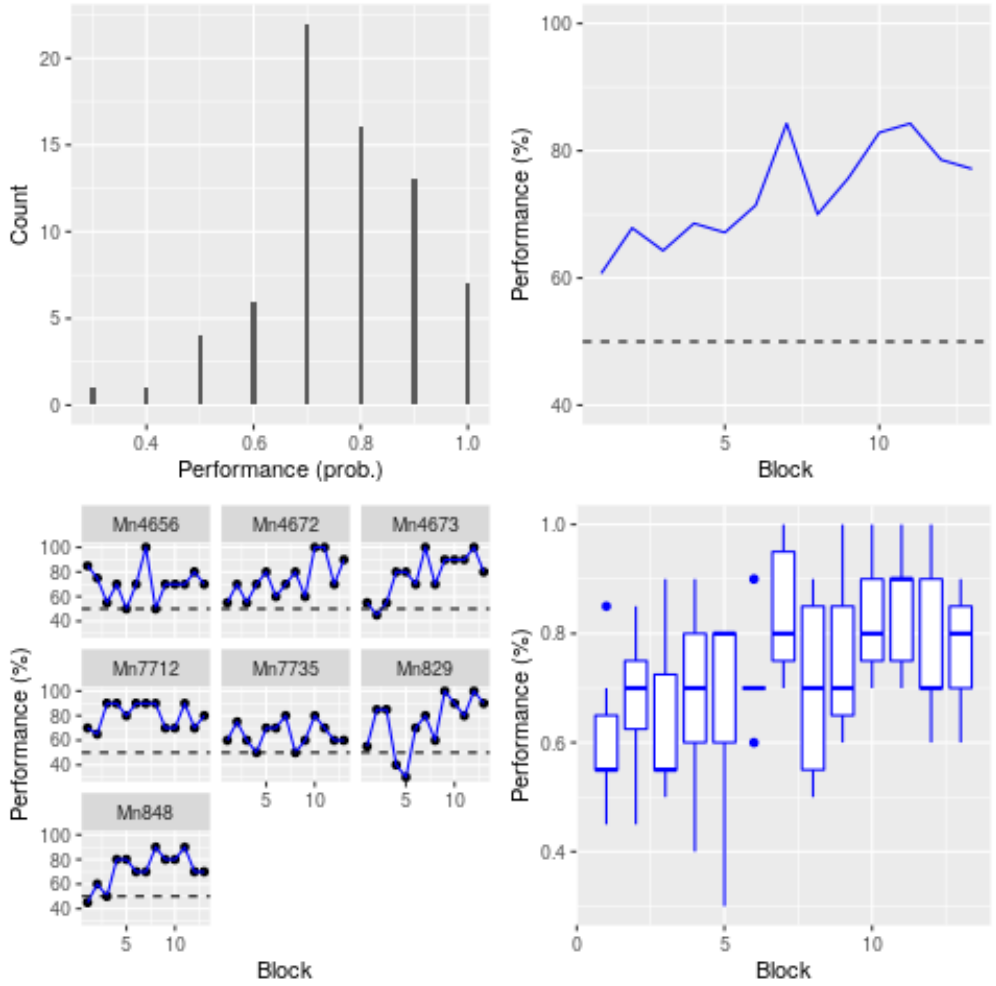
Kevin Allen

November 28, 2019

Today's plan

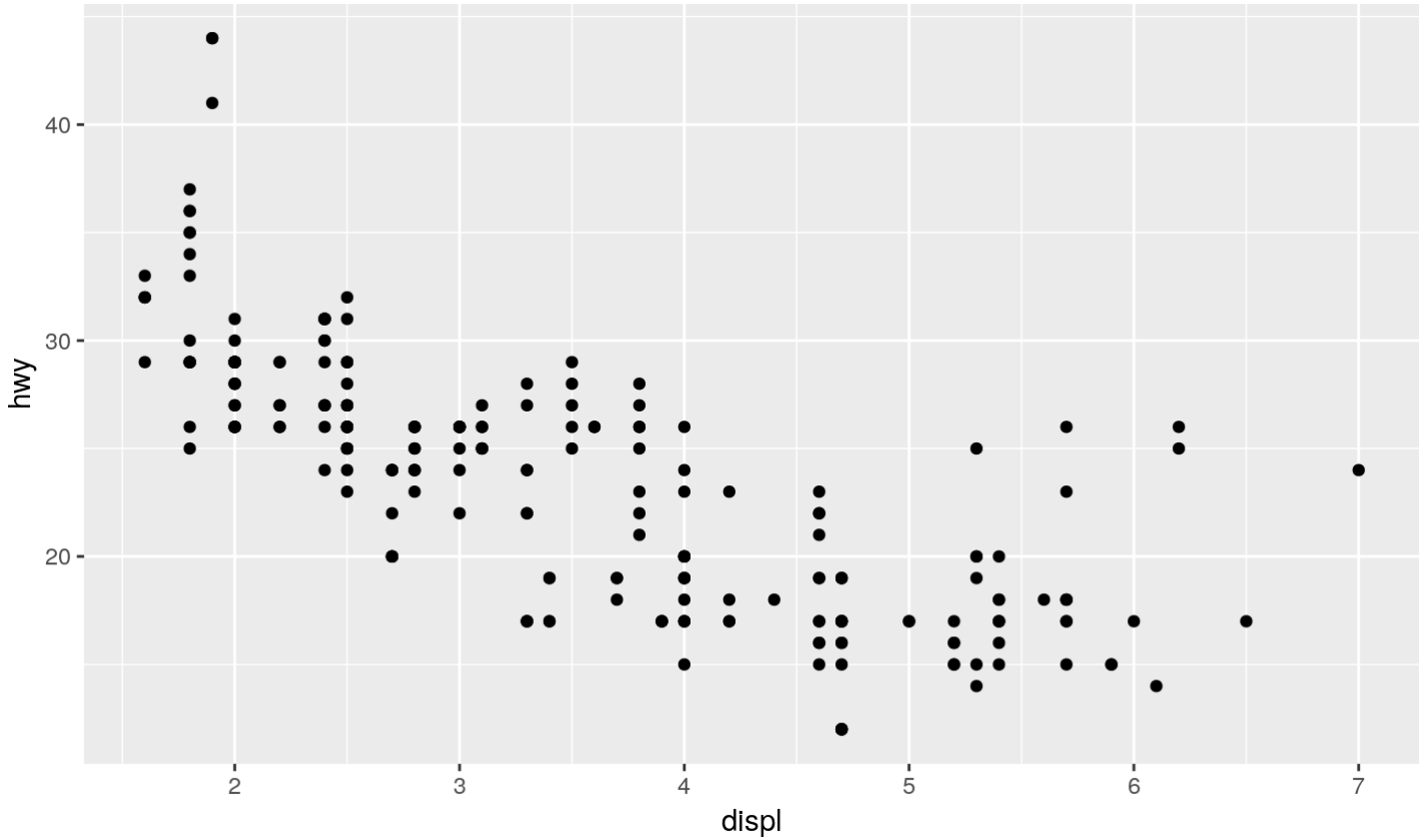
1. Review of last week
2. More exercises on dplyr and ggplot
3. Saving a graph
4. Plot with several graphs
5. git: create your git repository on github

Today's objective



Review: ggplot

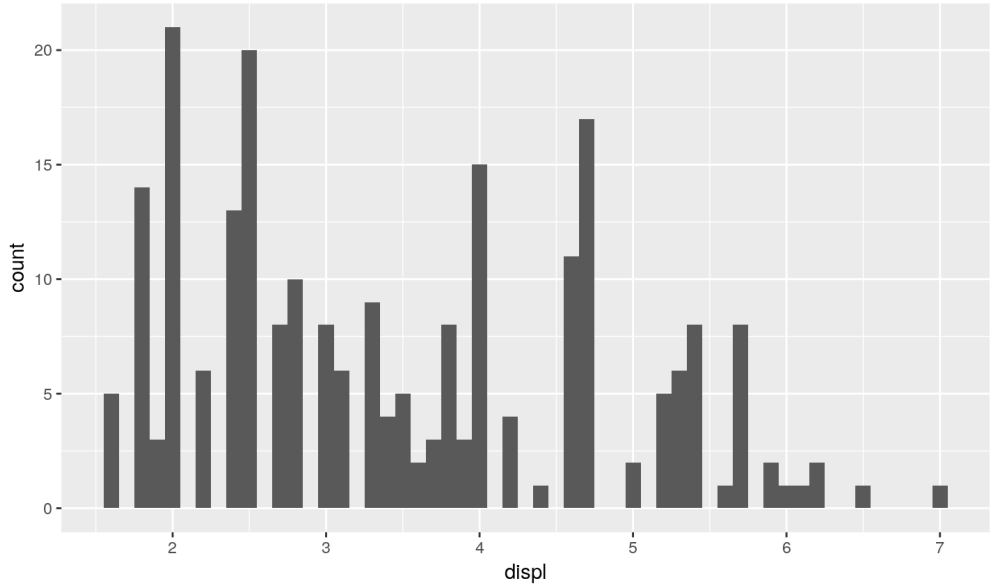
```
ggplot(data=mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



Count and distribution

Continuous variable

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



Review: 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()`

Review: load data from a 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()
## )
```

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

Review: load data from a file

```
df
```

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

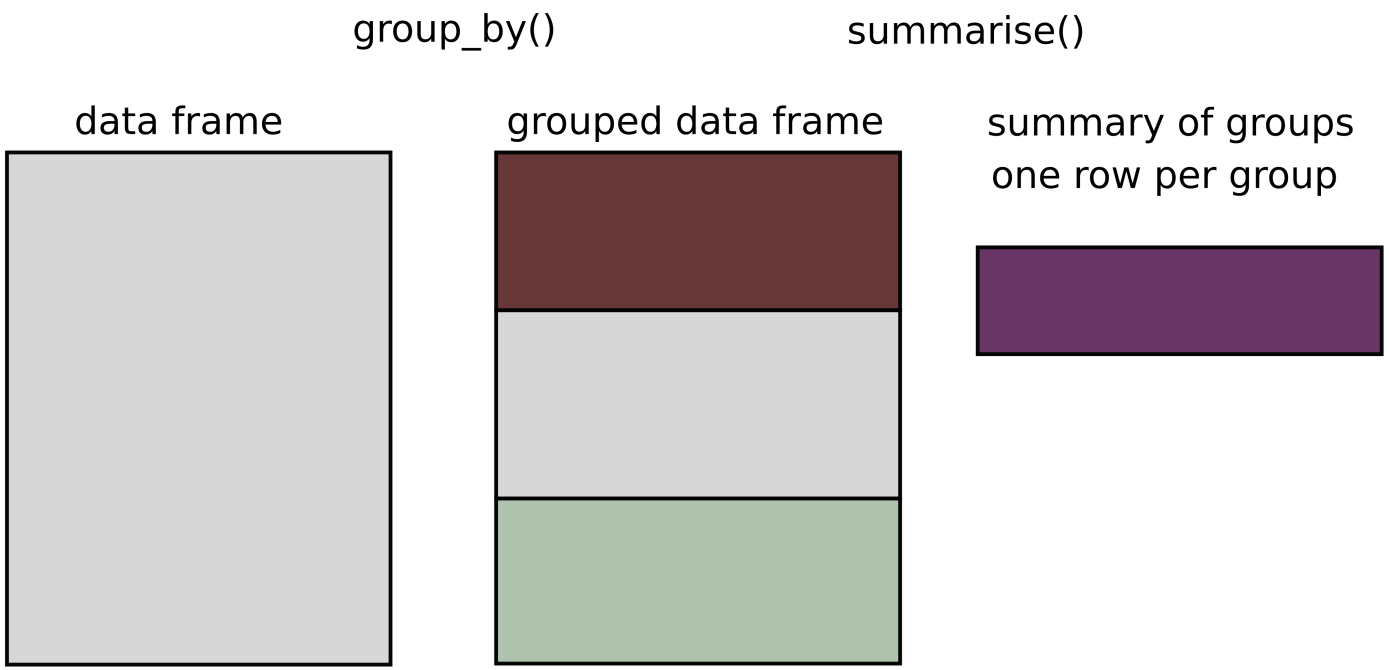
Review: summarise

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

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

Review: group_by and summarise

Collapse according to some groups.



Review: group_by and summarise

```
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
```

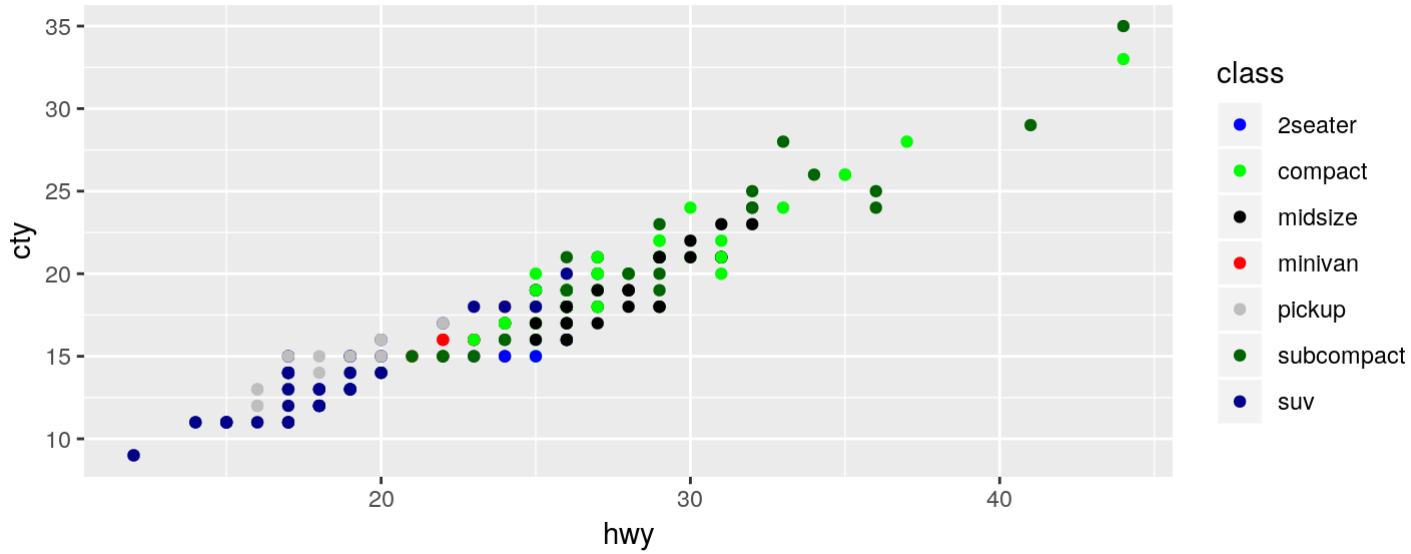
Review: using the pipe symbole (%>%)

```
blockdf <- df %>%
  select(mouse,block,correct) %>%
  group_by(block) %>%
  summarize(performance=mean(correct))
# print the first few lines
head(blockdf)
```

```
## # A tibble: 6 x 2
##   block performance
##   <dbl>       <dbl>
## 1     1       0.607
## 2     2       0.679
## 3     3       0.643
## 4     4       0.686
## 5     5       0.671
## 6     6       0.714
```

Question from last week

```
myColors<-c("2seater" = "blue", "compact" = "green",
            "midsize" = "black", "minivan" = "red",
            "pickup" = "grey", "subcompact" = "darkgreen",
            "suv" = "darkblue")
ggplot(data=mpg) +
  geom_point(mapping = aes(x=hwy,y=cty, color = class)) +
  scale_color_manual(values = myColors)
```



Work with dplyr and ggplot

- Create a data frame in which you have the performance of single mice on single blocks. Use the Pipe!
- Use the RStudio editor and save your code for later.

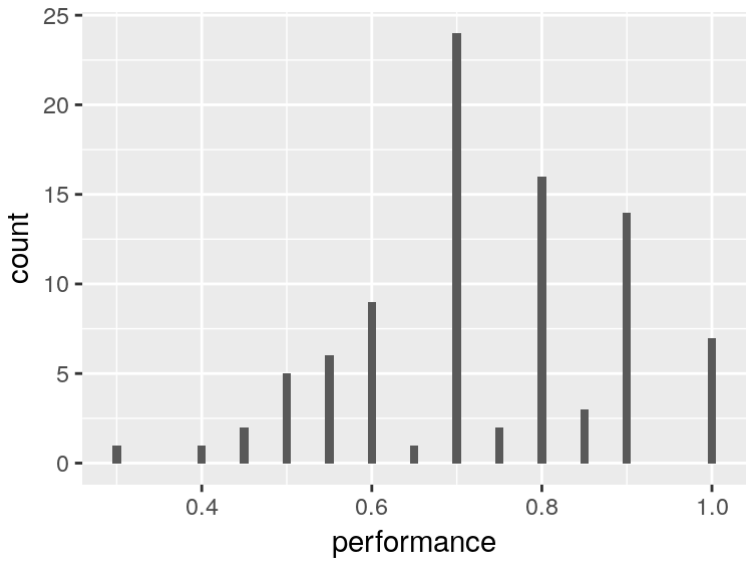
```
df %>%
  group_by(mouse,block) %>%
  summarise(performance = mean(correct))
```

```
## # A tibble: 91 x 3
## # Groups:   mouse [7]
##    mouse  block performance
##    <chr>  <dbl>       <dbl>
##  1 Mn4656     1        0.85
##  2 Mn4656     2        0.75
##  3 Mn4656     3        0.55
##  4 Mn4656     4        0.7
##  5 Mn4656     5        0.5
##  6 Mn4656     6        0.7
##  7 Mn4656     7        1
##  8 Mn4656     8        0.5
##  9 Mn4656     9        0.7
## 10 Mn4656    10        0.7
## # … with 81 more rows
```

Work with 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)
```

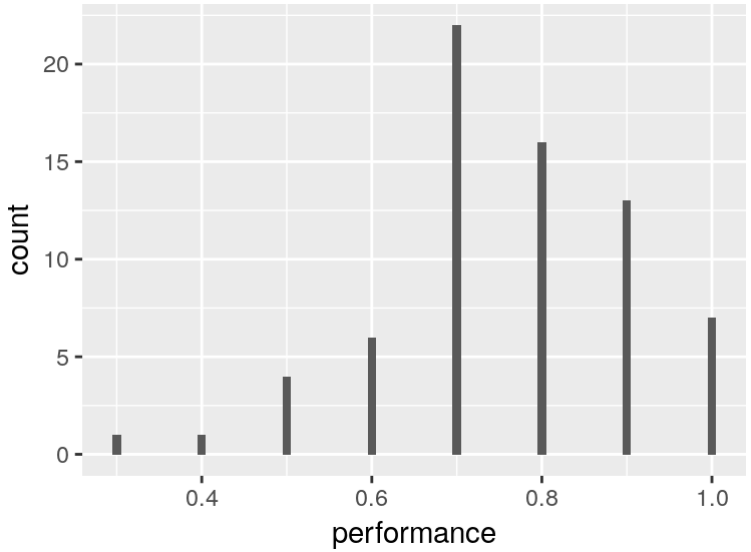


Work with dplyr and ggplot

- 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.

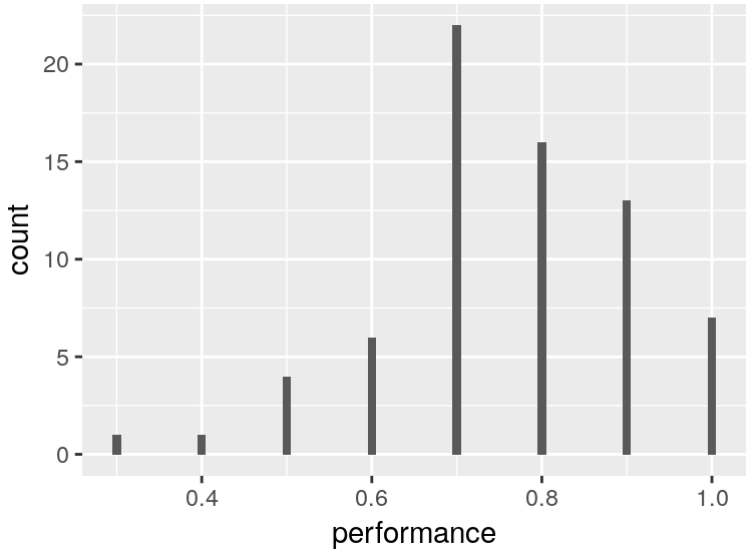
Solution 1

```
new_df <- df %>%
  group_by(mouse, block) %>%
  summarise(nTrials = n(), performance = mean(correct)) %>%
  filter(nTrials == 10) ## filter after summarise

ggplot (data = new_df) +
  geom_histogram(mapping = aes(x = performance), binwidth = 0.01)
```



Solution 2

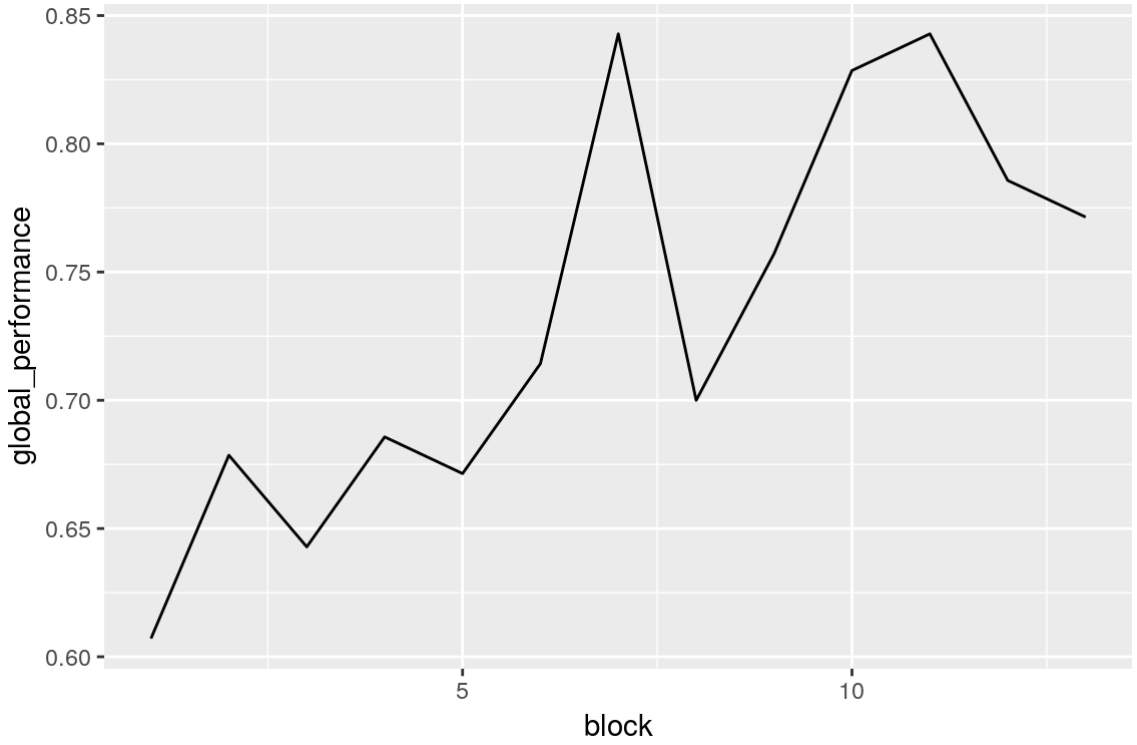
```
new_df <- df %>%
  filter(block > 3) %>% ## filter before summarise
  group_by(mouse, block) %>%
  summarise(performance = mean(correct))

ggplot (data = new_df) +
  geom_histogram(mapping = aes(x = performance), binwidth = 0.01)
```



Work with dplyr and ggplot

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

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

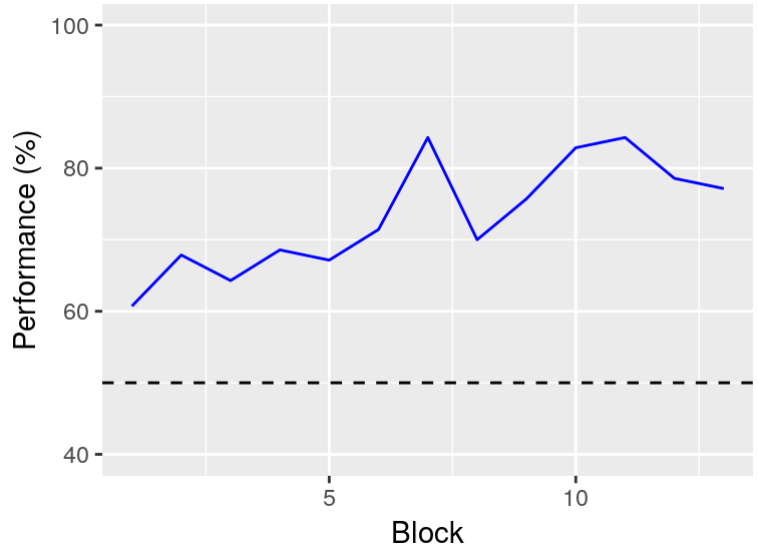


Work with dplyr and ggplot

Your supervisor is very picky. They want the following changes made to the last graph.

1. The performance should be in percentages instead of probability.
2. The line should be blue.
3. The x and y labels should be called "Performance (%)" and "Block", respectively
4. The range of the y axis to go from 40 to 100.
5. A horizontal line should be added at 50 % to indicate chance level.

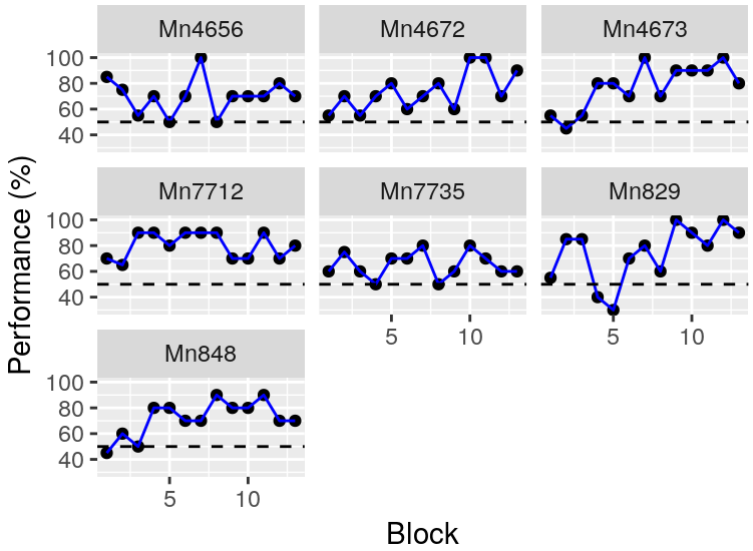
```
df %>% group_by(mouse,block) %>%
  summarise(performance = mean(correct)*100) %>%
  group_by(block) %>%
  summarise(global_performance = mean(performance)) %>%
  ggplot() +
  geom_line(mapping=aes(x=block,y=global_performance),color="blue")+
  ylim(40,100) +
  xlab("Block") +
  ylab("Performance (%)")+
  geom_hline(mapping = aes(yintercept=50),linetype="dashed")
```



Work with dplyr and ggplot

- Do you think all mice learned equally? It would be great to have a plot for each mouse. Use the facet_wrap function to plot several graphs into the same plot.

```
df1 <- df %>% group_by(mouse,block) %>%
  summarise(performance = mean(correct)*100)
ggplot(data=df1) +
  geom_point(mapping=aes(x=block,y=performance)) +
  geom_line(mapping=aes(x=block,y=performance),color = "blue") +
  ylab("Performance (%)") +
  xlab("Block") +
  geom_hline(mapping = aes(yintercept=50),linetype="dashed") +
  facet_wrap(~mouse)
```

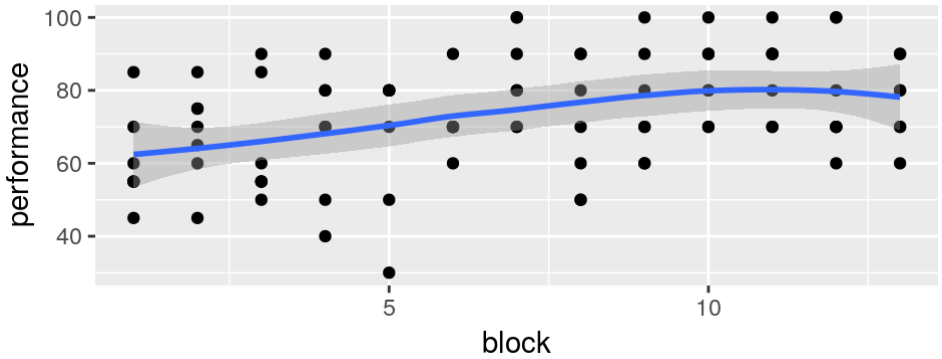


Work with dplyr and ggplot

- It would be great to have a graph that displays this variability in performance without having to show the 7 mice separately. How could we do this?

```
#df1 is the data frame we just created
df1 %>%
  ggplot() +
  geom_point(mapping=aes(x=block,y=performance))+
  geom_smooth(mapping=aes(x=block,y=performance))
```

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

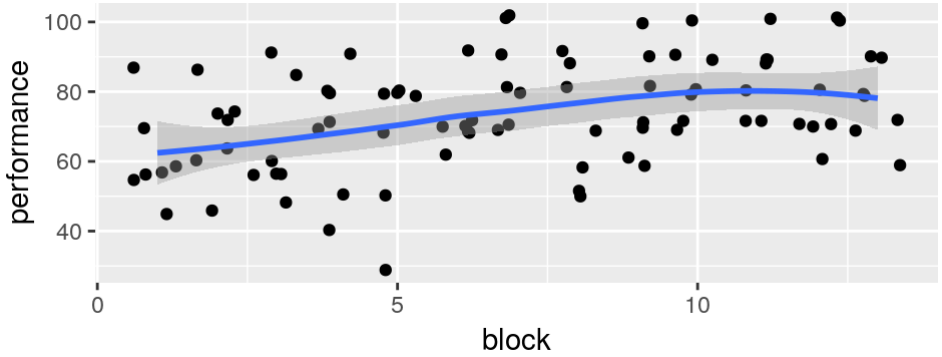


Do you see any problems with this plot?

Use some jitter to solve overplotting. This solve the overplotting but is less precise!

```
df1 %>%
  ggplot() +
  geom_point(mapping=aes(x=block,y=performance),position="jitter")+
  geom_smooth(mapping=aes(x=block,y=performance))
```

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

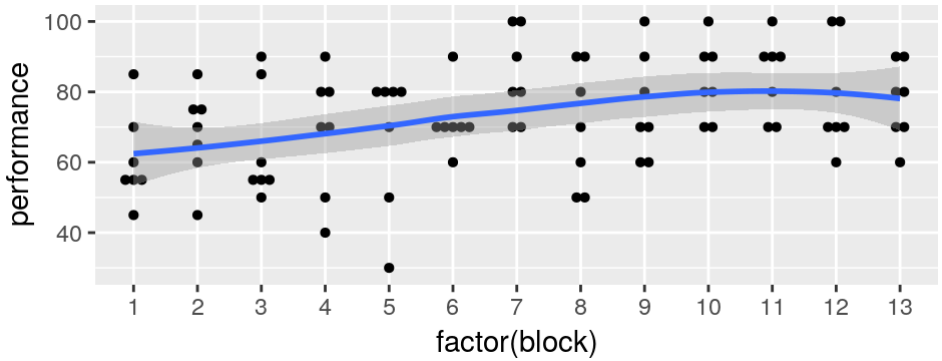


Most people will not like this solution. The block separation is gone.

```
df1 %>%
ggplot( aes(x = factor(block), y = performance)) +
  geom_dotplot(binaxis = "y", stackdir = "center") +
  geom_smooth(mapping=aes(x=block,y=performance))
```

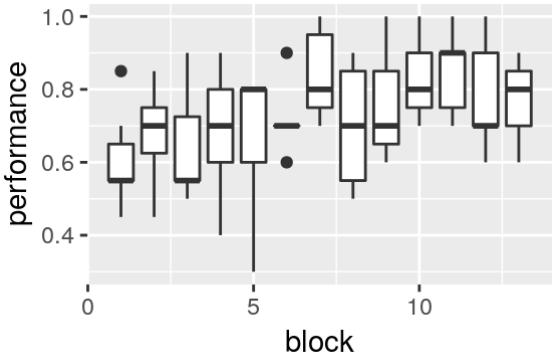
```
## `stat_bindot()` using `bins = 30`. Pick better value with `binwidth`.
```

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



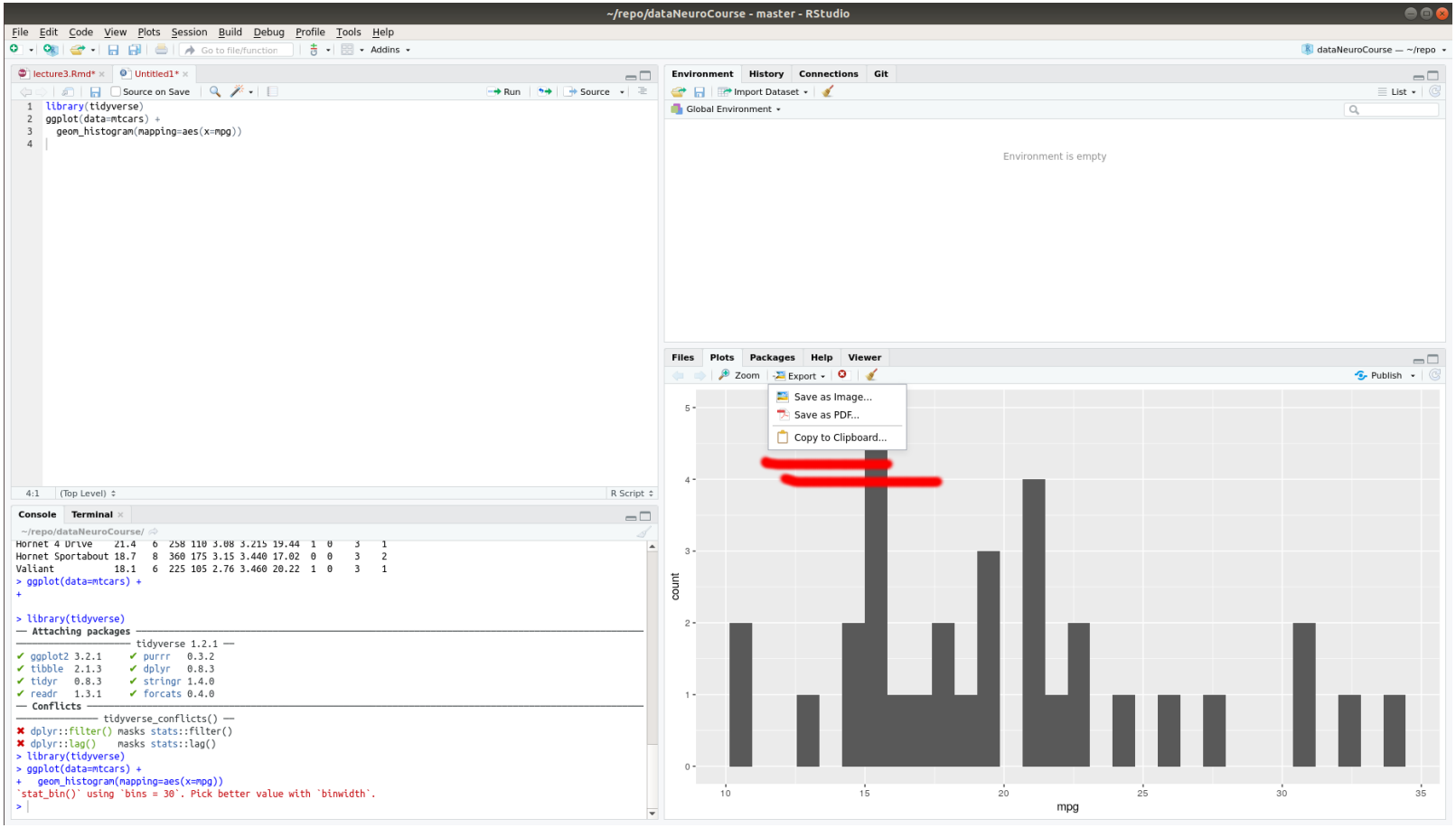
Each point now clearly belongs to a block. This is pretty good.

```
df1 <- df %>% group_by(mouse,block) %>%
  summarise(performance = mean(correct))
df1 %>%
  ggplot() +
  geom_boxplot(mapping=aes(x=block,y=performance,group=block))
```



This is also pretty good.

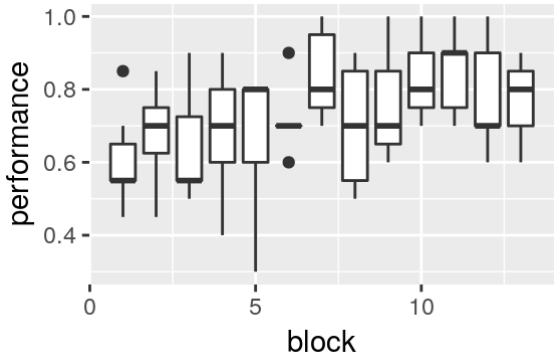
Saving a graph



Saving a graph

`ggsave()` saves the latest plot that was displayed

```
df1 %>%
  ggplot() +
  geom_boxplot(mapping=aes(x=block,y=performance,group=block))
```



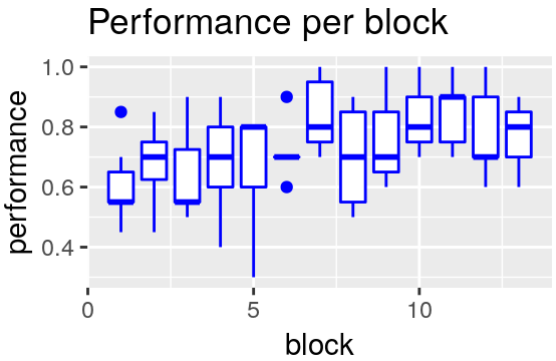

```
myFileName="/home/kevin/Downloads/myPlot.pdf"
ggsave(filename = myFileName,
       device = "pdf", units = "cm",
       width = 10, height = 10)
```

Plot with several graphs

- Store graphs as variables

```
p1<- df1 %>%
  ggplot() +
  geom_boxplot(mapping=aes(x=block,y=performance,group=block),color="blue")+
  ggtitle("Performance per block")
```

```
p1
```



Plot with several graphs

Let's store several graphs in variables

```
p1 <- df %>%
  group_by(mouse, block) %>%
  summarise(nTrials = n(), performance = mean(correct)) %>%
  filter(nTrials == 10) %>%
  ggplot () +
    geom_histogram(mapping = aes(x = performance), binwidth = 0.01) +
    xlab("Performance (prob.)") +
    ylab("Count")
```

```
p2 <- df %>% group_by(mouse,block) %>%
  summarise(performance = mean(correct)*100) %>%
  group_by(block) %>%
  summarise(global_performance = mean(performance)) %>%
  ggplot() +
  geom_line(mapping=aes(x=block,y=global_performance),color="blue")+
  ylim(40,100) +
  xlab("Block") +
  ylab("Performance (%)")+
  geom_hline(mapping = aes(yintercept=50),linetype="dashed")
```

```
p3 <-df %>% group_by(mouse,block) %>%
  summarise(performance = mean(correct)*100) %>%
  ggplot() +
    geom_point(mapping=aes(x=block,y=performance)) +
    geom_line(mapping=aes(x=block,y=performance),color = "blue") +
    ylab("Performance (%)") +
    xlab("Block") +
    geom_hline(mapping = aes(yintercept=50),linetype="dashed") +
    facet_wrap(~mouse)
```

```
p4 <- df %>% group_by(mouse,block) %>%
  summarise(performance = mean(correct)) %>%
  ggplot() +
  geom_boxplot(mapping=aes(x=block,y=performance*100,group=block),color="blue") +
  ylab("Performance (%)") +
  xlab("Block")
```

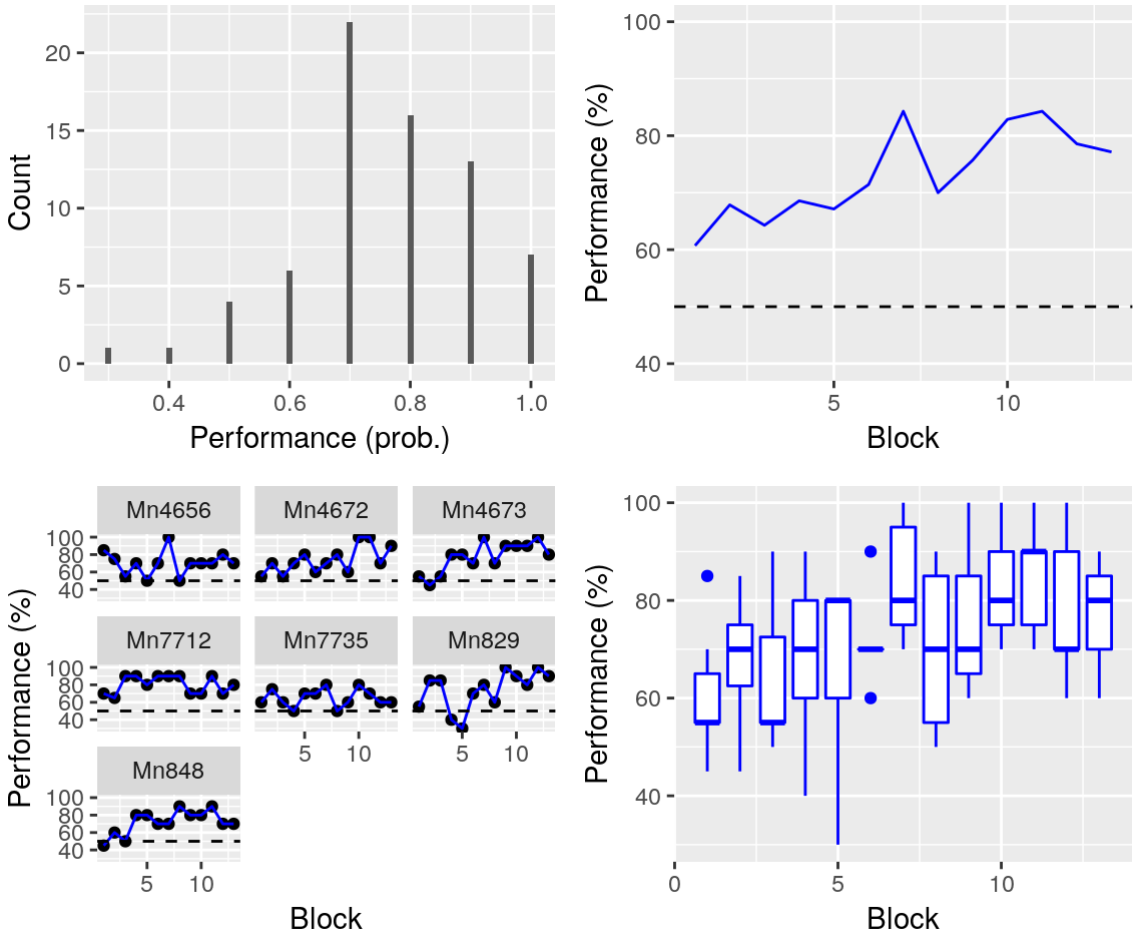
Now put these graphs together.

```
library(grid)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
grid.arrange(p1, p2, p3, p4, ncol=2)
```



Use `pdf()` and `dev.off()` to save your new creation.

```
myFileName="/home/kevin/Downloads/allMyPlots.pdf"
pdf(file=myFileName,paper = "a4") # open the file
all_ps <- grid.arrange(p1, p2, p3, p4, ncol=2) # write
dev.off() # close the file
```

```
## png
##   2
```

git: install

Install git on your computer

- Windows and Mac: (http://git-scm.com/downloads)
- linux: `sudo apt-get install git-core`

Example for Windows : Download for Windows, Run executable, license next, default location, will install git bash. Use Git and optional Unix tools from the Windows Command Prompt, 3 x next with default

Set up RStudio

1. Go to Global Options (from the Tools menu)
2. Click Git/SVN
3. Click Enable version control interface for RStudio projects

For more information

Create a repository on GitHub.

1. Create an account at GitHub and verify your email
2. Log in
3. Click New repository
4. Set repository name: myNotesDataScience, set to private, check Initialize this repository with a README
5. Click Create repository

Create a project with RStudio from your repository

1. Copy the urs of your repository from the github web site. It ends with `.git`
2. Open RStudio, New Project…, Version Control, Git,
3. Enter the url and set directory.
4. Create project.

Your first commit and push

1. Click on commit.
2. Make sure `myNotesDataScience.Rproj` is staged.
3. Set commit message to `my first commit`
4. Click commit
5. Click on the up arrow to push your commit to the online repository.
6. Refresh your web browser to see `myNotesDataScience.Rproj` online.

Next steps…

All you will have to do most of the time.

1. Save the file with your R code in your project directory.
2. Stage the file
3. Commit
4. Push