Introduction to R

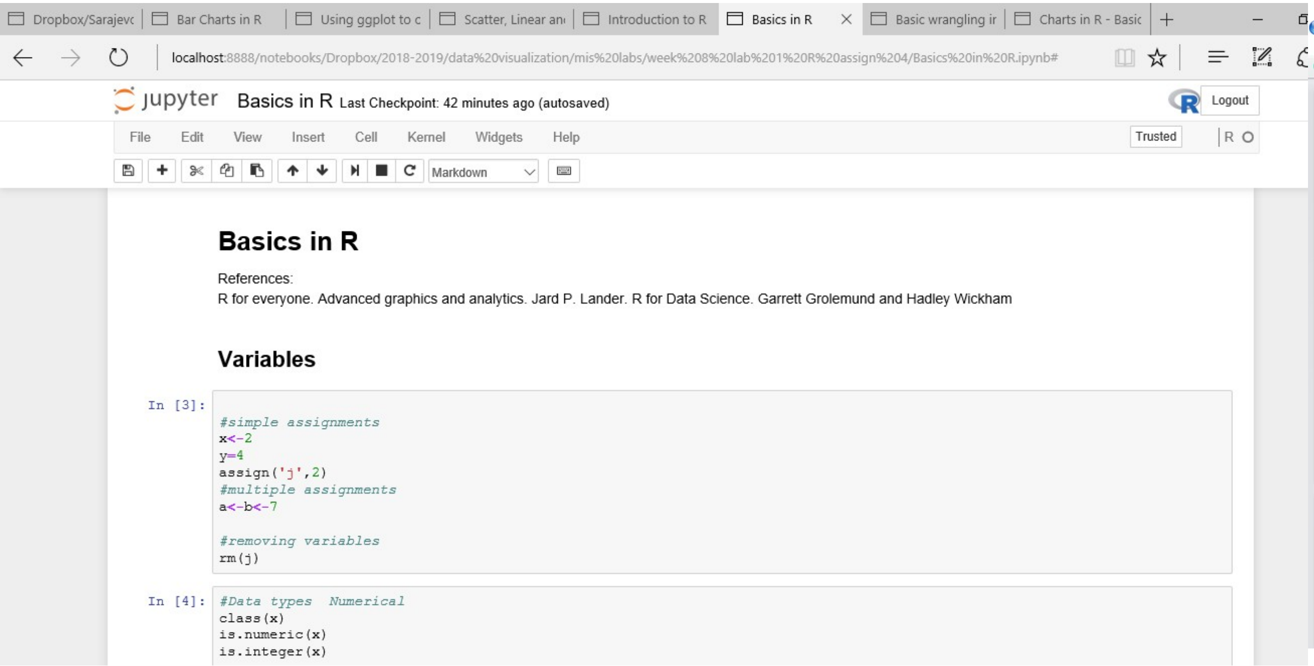
Cathy Ennis

Overview

- Introduction to R and environments
- Getting data
- Working with data
- Basic plots
- ggplot2

R and IDEs

- R
- Rstudio
- Jupyter Notebooks with R kernel



Basics in R

References:
R for everyone. Advanced graphics and analytics. Jard P. Lander. R for Data Science. Garrett Grolemund and Hadley Wickham

Variables

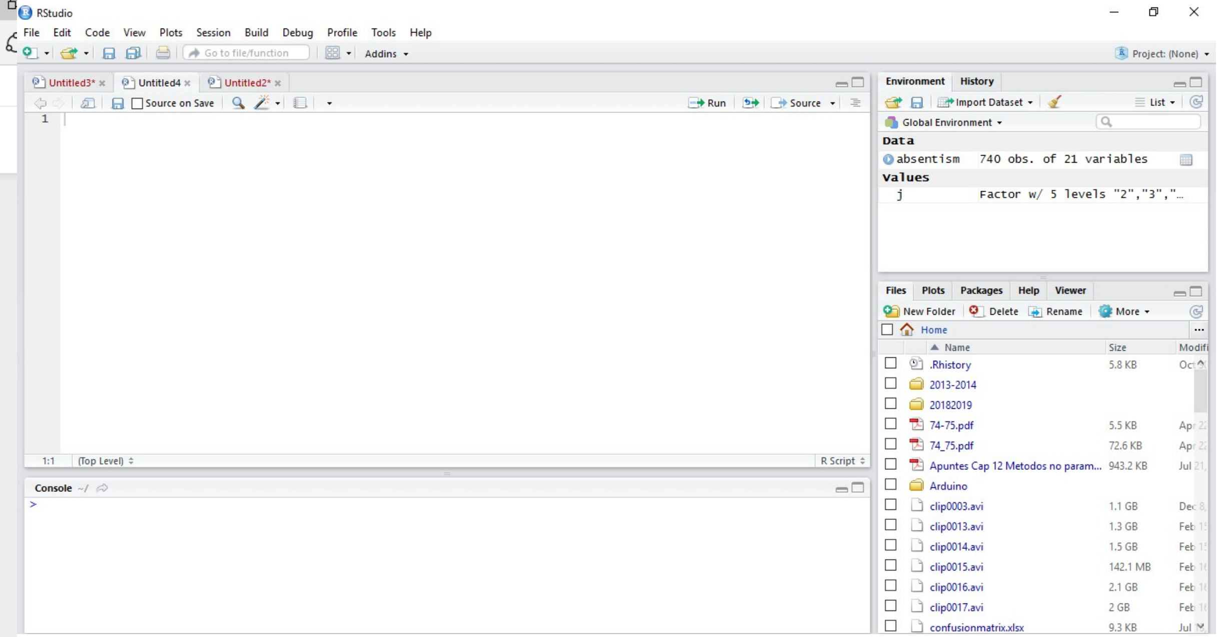
In [3]:

```
#simple assignments
x<-2
y=4
assign('j',2)
#multiple assignments
a<-b<-7

#removing variables
rm(j)
```

In [4]:

```
#Data types  Numerical
class(x)
is.numeric(x)
is.integer(x)
```



```
1
```

```
>
```

Data

absentism	740 obs. of 21 variables

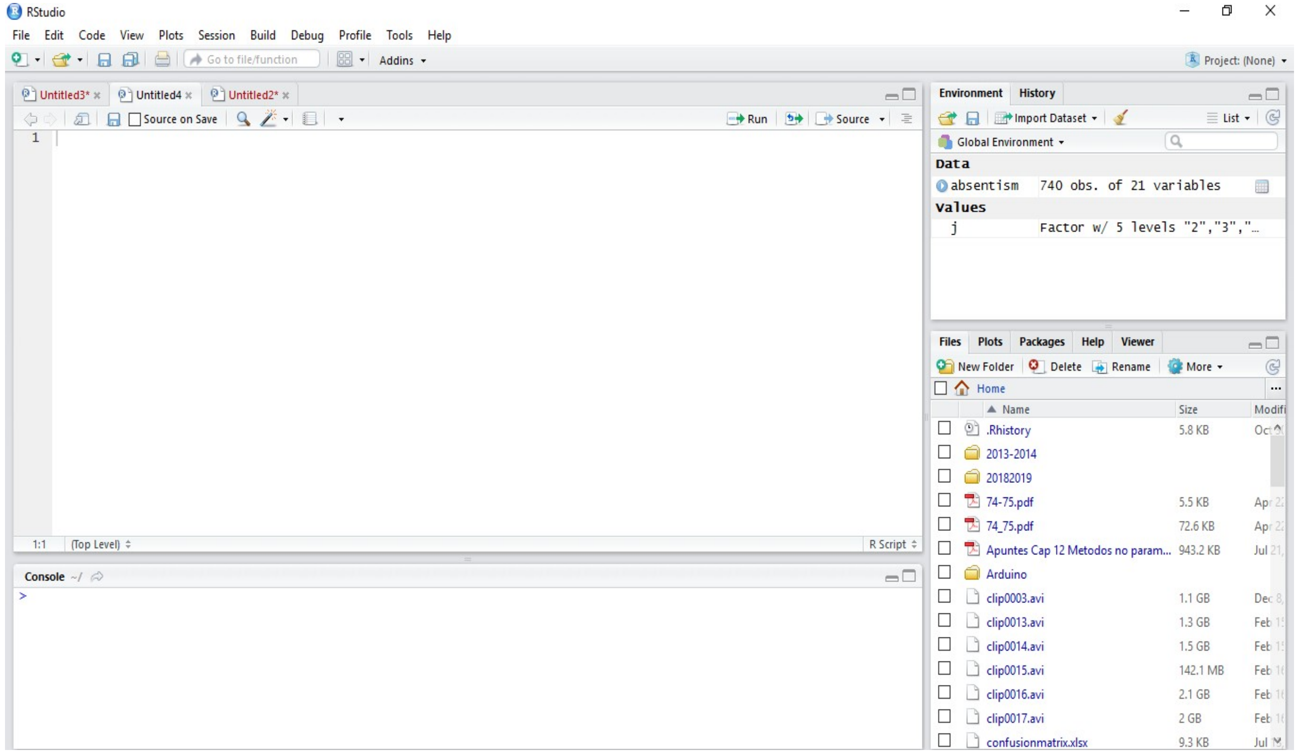
Values

j	Factor w/ 5 levels "2","3","…

Name	Size	Modifi
.Rhistory	5.8 KB	Oct
2013-2014		
20182019		
74-75.pdf	5.5 KB	Apr 2
74_75.pdf	72.6 KB	Apr 2
Apuntes Cap 12 Metodos no param...	943.2 KB	Jul 21,
Arduino		
clip0003.avi	1.1 GB	Dec 8,
clip0013.avi	1.3 GB	Feb 1
clip0014.avi	1.5 GB	Feb 1
clip0015.avi	142.1 MB	Feb 1
clip0016.avi	2.1 GB	Feb 1
clip0017.avi	2 GB	Feb 1
confusionmatrix.xlsx	9.3 KB	Jul

R and IDEs

- R
- Rstudio

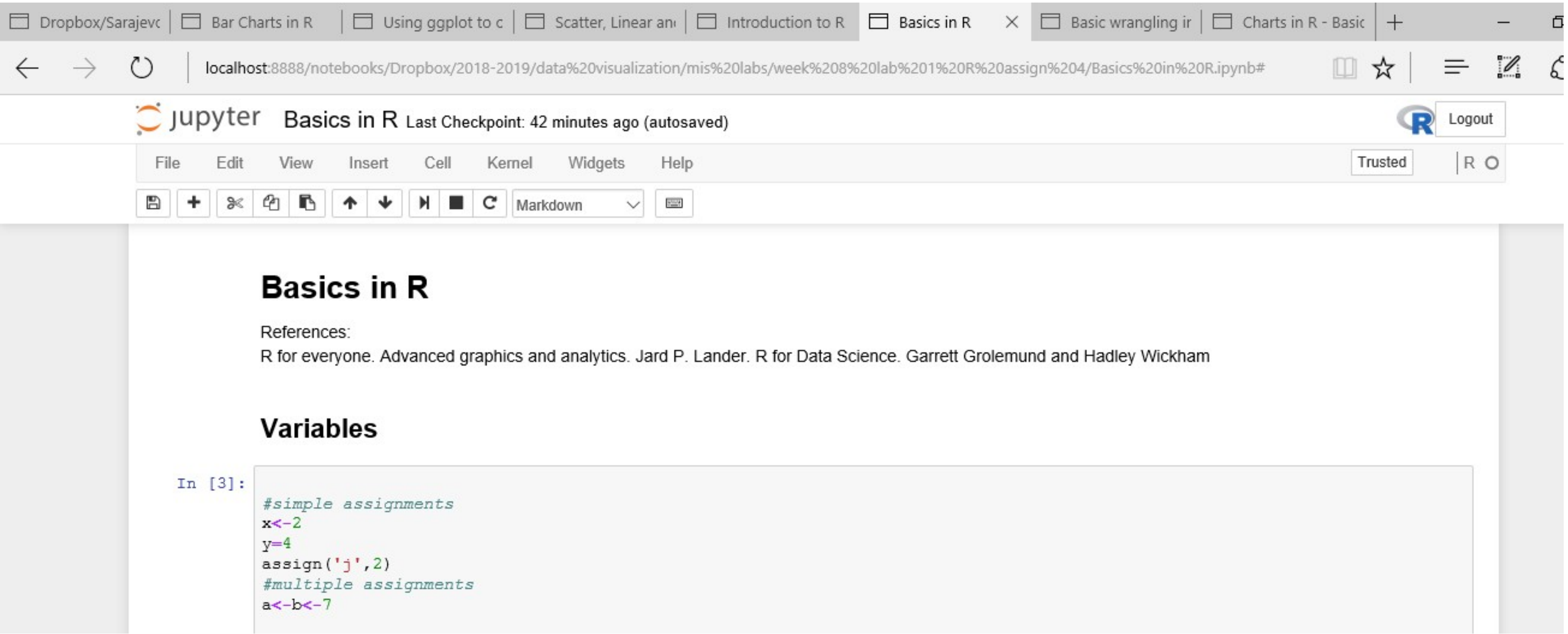


```
Data
absentism    740 obs. of 21 variables
Values
j            Factor w/ 5 levels "2","3","…
```

□	Name	Size	Modifi
□	.Rhistory	5.8 KB	Oct
□	2013-2014		
□	20182019		
□	74-75.pdf	5.5 KB	Apr 2
□	74_75.pdf	72.6 KB	Apr 2
□	Apuntes Cap 12 Metodos no param...	943.2 KB	Jul 21,
□	Arduino		
□	clip0003.avi	1.1 GB	Dec 8,
□	clip0013.avi	1.3 GB	Feb 1
□	clip0014.avi	1.5 GB	Feb 1
□	clip0015.avi	142.1 MB	Feb 1
□	clip0016.avi	2.1 GB	Feb 1
□	clip0017.avi	2 GB	Feb 1
□	confusionmatrix.xlsx	9.3 KB	Jul

R and IDEs

- R
- Rstudio
- Jupyter Notebooks with R kernel



Basics in R

References:
R for everyone. Advanced graphics and analytics. Jard P. Lander. R for Data Science. Garrett Grolemund and Hadley Wickham

Variables

In [3]:

```
#simple assignments
x<-2
y=4
assign('j',2)
#multiple assignments
a<-b<-7
```

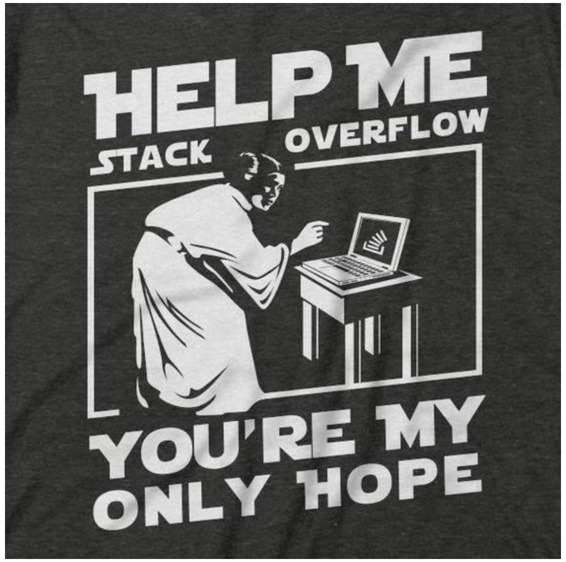
Need help?

- Help for functions in R
- ?function_name -> Description and examples
- Package documentation

Need extra help?

- Help for functions in R
- ?function_name -> Description and examples
- Package documentation

- Google it!
- Active Community
- Stack Overflow



Basics in R

Inspection of datasets/variables

- quickly view a dataset that has been loaded into Rstudio
- get more information about your data frame
- number of columns, number of rows, and length
- get the names of each column
- get the dimensions (number of rows and cols)

```
View(studentresult)
str(studentresult)
ncol(studentresult)
nrow(studentresult)
length(studentresult)
names(studentresult)
dim(studentresult)
```

Basics in R

The main data structures in R are:

- **list** - Contains data elements of any type
- **vector** - Sequence of data elements of the same basic type
- **matrix** - Vector with additional dimensions
- **data frame** - Used to store data tables, list of vectors of equal length

The data primitives in R are:

- **numeric** - Numeric data (approximations of the real numbers, $\mathbb{R}$)
- **integer** - Integer data (whole numbers, $\mathbb{Z}$)
- **factor** - Categorical data (simple classifications, like gender)
- **ordered** - Ordinal data (ordered classifications)
- **character** - Character data (strings)
- **raw** - Binary data

Basics in R

- Variable types
- Manipulating variables
- Slicing/accessing values within variables
  - vector3<-c("a","b","ccc","dd","eeee") vector3[1]
  - vector3[2:3] vector3[c(2,5)] vector3[-3]
- Advanced data structures
  - Data frames
- Reading data adult<-read.table(…
- Vectorized calculations
  - length(which(adult$occupation == " ?"))

```
#simple assignments
x<-2
y=4
assign('j',2)

#multiple assignments
a<-b<-7

#removing variables
rm(j)

 #Data types  Numerical
 class(x)
 is.numeric(x)
 is.character(x)
```

```
# #Data types  Characters
 xchar<- 'data'
 nchar(xchar)
 ychar<-factor("Data")


# #dates
 date1<-as.Date("2018-09-09")
 class(date1)
 date2<-as.Date("2018-09-02")
 date1>date2
```

Basics in R

- Variable types
- Manipulating variables
- Slicing/accessing values within variables
  - vector3<-c("a","b","ccc","dd","eeee") vector3[1]
  - vector3[2:3] vector3[c(2,5)] vector3[-3]
- Advanced data structures
  - Data frames
- Reading data adult<-read.table(…
- Vectorized calculations
  - length(which(adult$occupation == " ?"))

```
##creating a vector using c
vector1<-c(1,2,3,4,5,5)
##creating a vector using : (sequence of numbers)
vector2<-1:6
##length of avector
length(vector1)
##Vector operations
vector1*3
vector1+vector2
#length ofthis addtiion should be 6 as well, test it!
#no needs for loops
#Comparing vectors to values
vector1<5
any(vector1<5)
all(vector1<5)
```

```
##using nchar with vectors
vector3<-c("a","b","ccc","dd","eeee")
nchar(vector3)
vector3[1]
vector3[2:3]
vector3[c(2,5)]
vector3[-3]
###different to python CAREFUL!!!!
```

Basics in R

- Variable types
- Manipulating variables
- Slicing/accessing values within variables
  - vector3<-c("a","b","ccc","dd","eeee") vector3[1]
  - vector3[2:3] vector3[c(2,5)] vector3[-3]
- Advanced data structures
  - Data frames
- Reading data adult<-read.table(…
- Vectorized calculations
  - length(which(adult$occupation == " ?"))

```
##Factor vectors
g<-c("January","February","March","April","May","June","June")
g
gfactor<-as.factor(g)
levels(gfactor)
```

Basics in R

- Variable types
- Manipulating variables
- Slicing/accessing values within variables
  - vector3<-c("a","b","ccc","dd","eeee") vector3[1]
  - vector3[2:3] vector3[c(2,5)] vector3[-3]
- Advanced data structures
  - Data frames
- Reading data adult<-read.table(…
- Vectorized calculations
  - length(which(adult$occupation == " ?"))

```
x<-1:10
y<-4:13
comments<-c("good", "basic", "Excellent","good",
"basic","good",
"basic","Excellent","good","Excellent")
mydf<-data.frame(x,y,comments)
mydf
##naming our columns
mydf<-
data.frame(First=x,Second=y,Opinion=comments)
mydf
##naming our columns afterwards
colnames(mydf) <- c("First","Second","Third")
mydf
names(mydf)
names(mydf)[2]
head(mydf)
tail(mydf)
class(mydf)
table(mydf$Third)
```

```
## #Accessing one column
mydf$Opinion
mydf[2]
# #Accessing one row
mydf[3,]

# #accessing one element
mydf[1,3]
```

Basics in R

- Variable types
- Manipulating variables
- Slicing/accessing values within variables
  - vector3<-c("a","b","ccc","dd","eeee") vector3[1]
  - vector3[2:3] vector3[c(2,5)] vector3[-3]
- Advanced data structures
  - Data frames
- Reading data adult<-read.table(…
- Vectorized calculations
  - length(which(adult$occupation == " ?"))

```
# #Reading csv files
# read.table creates a data.frame with the data.  ## #From URL
mydata1<-
read.table("http://www.jaredlander.com/data/Tomato%20First.csv",heade
r=TRUE,sep=',')
head(mydata1)


mydata2<-
read.csv2("http://www.jaredlander.com/data/Tomato%20First.csv",heade
r=TRUE,sep=',')
head(mydata2)


adult<-read.table('https://archive.ics.uci.edu/ml/machine-learning-
databases/adult/adult.data',header=FALSE,sep=',’)

dim(adult)  colnames(adult) <-
c("age","workclass","fnlwgt","education","education_num","marital_statu
s","occupation","relationship","race","gender","capital_gain","capital_loss
","hours_per_week","native_c  ountry","income_bracket")

head(adult)
```

Basics in R

- Variable types
- Manipulating variables
- Slicing/accessing values within variables
  - vector3<-c("a","b","ccc","dd","eeee") vector3[1]
  - vector3[2:3] vector3[c(2,5)] vector3[-3]
- Advanced data structures
  - Data frames
- Reading data adult<-read.table(…
- Vectorized calculations
  - length(which(adult$occupation == " ?"))

```
length(which(adult$occupation == " ?"))
any(is.na(adult$gender))
summarygender<-table(adult$gender)
summarygender[1]
summarygender[2]

aggregate(adult$age~adult$gender,adult,mean)
aggregate(adult$age~adult$gender,adult,sum)
aggregate(adult$age~adult$gender,adult,median)
?aggregate
```

Sqldf library

- Use standard SQL functions on a data frame
- sqldf allows us to run sql queries across data frames enabling us to do things like group by and sophisticated date queries.
- One of the most popular queries for cubed-type data is a group by.

To run a simple group by query in R and SQLDF, run the following command:

```
marks <- sqldf(‘select subject, sum(Mark_Written)
as mark from studentresult  group by subject’)
View(marks)
```

This shows the marks per student.

Without sqldf, using aggregate:

```
markswritten2<-aggregate(data=studentresult,
Mark_Written~ Name,mean)
```

Sqldf library - Impute missing values

- Imputing missing values
- There are several ways to do this, in this example we will average scores in Oral exams over the previous two years and then use that average as the score for the missing year

The following command selects the average for Mary Healy where there are no NAs:

```
avgmark <- sqldf ( “ select ROUND(AVG(Mark_Oral) )
from studentresult where Name = ‘Mary  Healy' AND
Mark_Oral is not ‘NA‘ ")
```

Version 2 without sqldf:

```
avgmark2<- mean( studentresult$Mark_Oral [ !is.na(
studentresult$Mark_Oral) & studentresult$Name ==
'Mary Healy' ] )
```

Replaces those NA with the average from the query:

```
studentresult$Mark_Oral <- ifelse(
is.na(studentresult$Mark_Oral), as.numeric(avgmark),
studentresult$Mark_Oral)
```

Charts in R

- **ggplot**
  - built on the idea that you build every graph from the same components, a data set, a set of geoms and a coordinate system.
  - Layers of information can be added to customize your visualisation.

- ggplot structure

```
myplot <- ggplot(data= yourdataset, aes(x=yourx, y = youry))
# this begins your plot by adding the data
```

Charts in R

```
myplot +geom_point() # this adds a geometry to your plot (scatter plot in the example)

myplot +geom_point(aes(color=dimension)    # geom layer can be customized

myplot +geom_bar()+scale_fill_brewer(palette=’Reds’)    # customizing the colour palette

myplot +geom_bar(color=dimension) + scale_fill_manual(values=c(‘blue’,’red’))   #customize colours manually

myplot+coord_map(projection=”ortho”, orientation= c(41,-74,0))      # map projection

myplot + theme_classic()              # applies a predefined theme to the plot

myplot + labs(title=”graph title”, x=”xaxis title”, y=”yaxistitle”)

myplot + facet_wrap(dimension)   # creates small multiples based on dimension
```