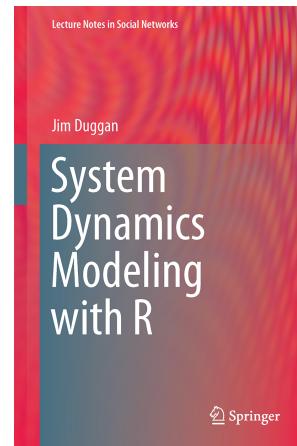


Part I

Introduction to R



Dr. Jim Duggan,
School of Engineering & Informatics
National University of Ireland Galway.

<https://github.com/JimDuggan/SDMR>

https://twitter.com/_jimduggan



NUI Galway
OÉ Gaillimh

System Dynamics Modeling with R

© Jim Duggan 2016

First step... download materials...

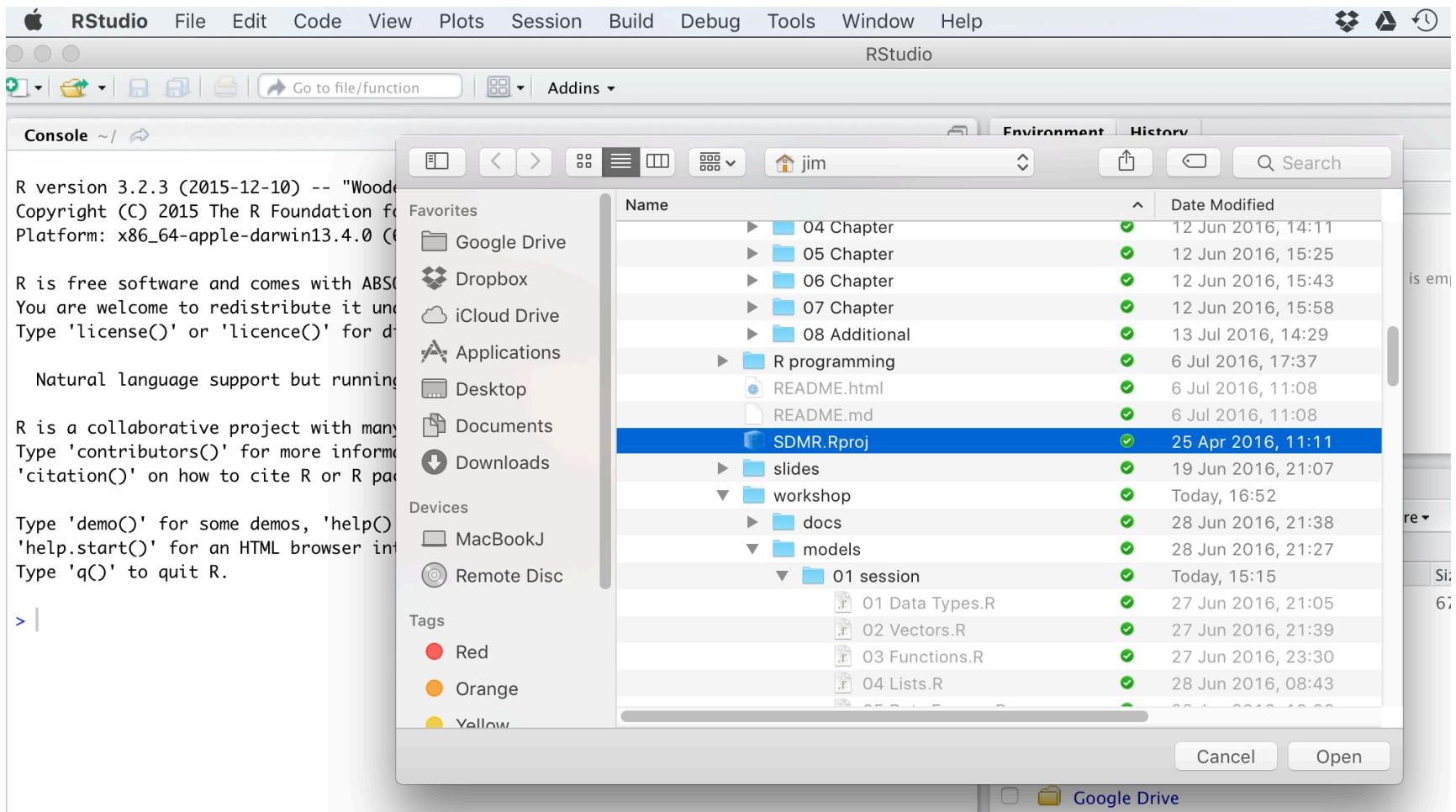
<https://github.com/JimDuggan/SDMR>

The screenshot shows a Safari browser window with the GitHub repository page for "JimDuggan / SMDR". The repository name is "JimDuggan / SMDR". The page displays basic statistics: 48 commits, 1 branch, 0 releases, and 1 contributor. A list of recent commits is shown, including updates to solution files, R programming examples, conference paper examples, and README files. On the right side of the commit list, there is a "Clone with SSH" section with a copy link and options to "Open in Desktop" or "Download ZIP".

Commit	Description	Date
JimDuggan Updating solution file...	Adding conference paper example - Bass Mod...	26 days ago
R programming/examples/ggplot2		
images	First draft of README summary	
models	New directory...	
slides/system dynamics	Updating slide folder structure	
workshop	Updating solution file...	3 hours ago
.gitignore	Test file	3 months ago
README.html	Update on workshop README	9 days ago
README.md	Minor edits...	a month ago
SDMR.Rproj	Test file	3 months ago



Open Project <SDMR.Rproj>



Initial Screen

The screenshot shows the RStudio interface with the following components:

- Top Bar:** RStudio, File, Edit, Code, View, Plots, Session, Build, Debug, Tools, Window, Help.
- Toolbar:** Go to file/function, Addins.
- Environment Tab:** Environment, History, Git. The Global Environment pane shows "Environment is empty".
- Files Tab:** Files, Plots, Packages, Help, Viewer. The Files pane shows the project structure:

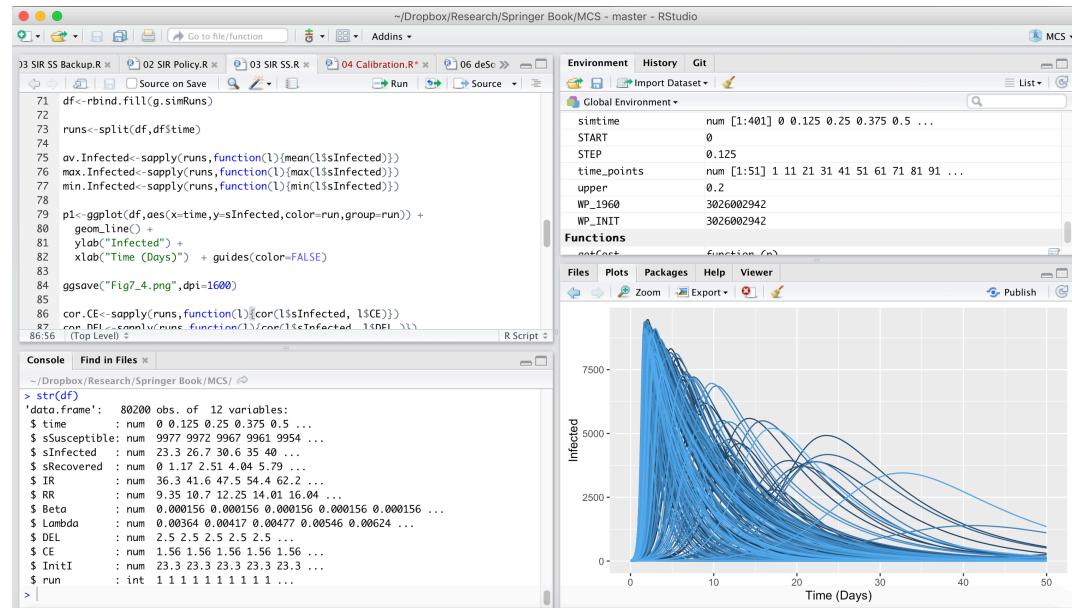
Name	Size	Modified
.gitignore	40 B	Apr 25, 2016, 11:11 AM
.RData	37.5 KB	May 2, 2016, 9:29 PM
.Rhistory	17.4 KB	Jul 15, 2016, 6:23 PM
images		
models		
R programming		
README.html	834.8 KB	Jul 6, 2016, 11:08 AM
README.md	6.8 KB	Jul 6, 2016, 11:08 AM
SDMR.Rproj	205 B	Apr 25, 2016, 11:11 AM
slides		
workshop		

- Console:** ~ /Dropbox/R Projects/SDMR/
- Text Editor:** README.md (Markdown)
- Help:** Natural language support but running in an English locale
- R Information:** R is a collaborative project with many contributors. Type 'contributors()' for more information and 'citation()' on how to cite R or R packages in publications.
- Help:** Type 'demo()' for some demos, 'help()' for on-line help, or 'help.start()' for an HTML browser interface to help.
- Quit:** Type 'q()' to quit R.



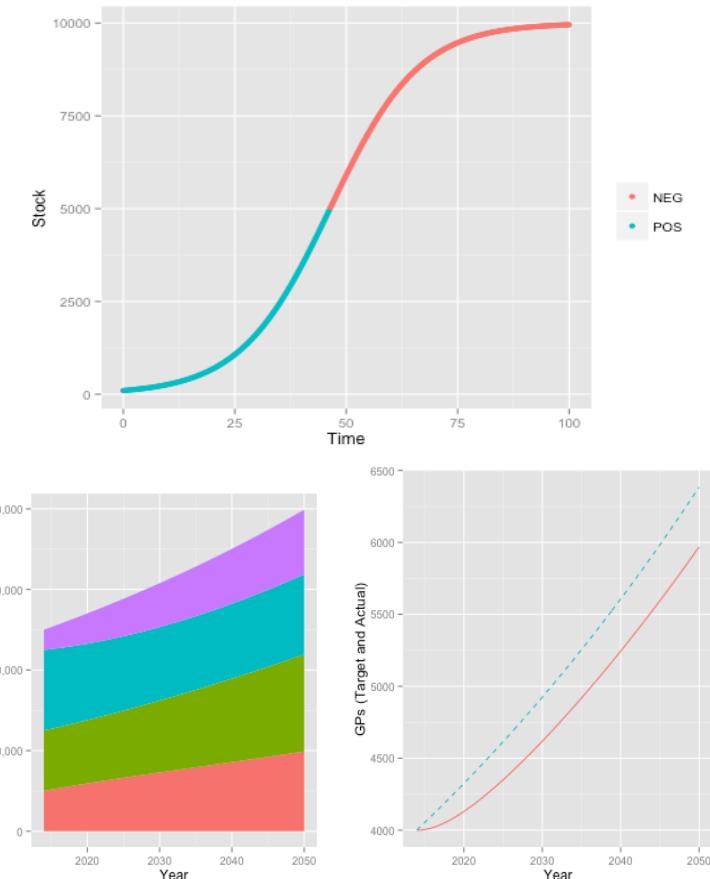
Outline

- Introduction to R
- Storing data
 - Vectors
- Functions
- Storing data
 - Lists
 - Data Frames
- Visualisation



What is R?

- It is a dialect of the S language, developed at Bell Laboratories
- Open source, functional & object-oriented language
- R's *mission* is to enable the best and most thorough exploration of data possible (Chambers 2008).



R Studio Workbench

R Code

Interactive console

Environment/State

The screenshot shows the RStudio interface with a red border around the bottom-left pane.

Global Environment:

Value	Type	Content
b1	logi	[1:2] FALSE TRUE
b2	logi	[1:5] FALSE TRUE FALSE TRUE FALSE
c1	chr	[1:5] "Odd" "Even" "Odd" "Even" "Odd"
ind	2L	
index	5L	
props	table [1:3(1d)]	0.4 0.3 0.3
r		24
s	num	[1:101] 51.4 55 57.2 57.3 58.6 ...
s1	int	[1:20] 2 1 1 3 1 3 2 1 1 1 ...
s2	int	[1:20] 2 3 2 2 2 2 2 1 2 2 1 ...

Files:

Name	Size	Modified
..		
01 Introduction.R	1.9 KB	Sep 3, 2015, 2:23 PM
01 Vectors.pdf	892 KB	Sep 3, 2015, 2:26 PM

File System



(1) Data Structures - Vector

- The fundamental data type in R is the vector,
- A variable that contains a sequence of elements that have the same data type (Matloff 2009).
- Create using $c(e_1, \dots, e_n)$
- Assignment statement $<-$

v1

1
4
9
16
25

```
> v1<-c(1,4,9,16,25)
> v1
[1] 1 4 9 16 25
```



Four main types of atomic vectors

- logical
- integer
- double
- character

```
> v5<-c(v1,v2)
> v5
[1] 1 0 10 20

> typeof(v5)
[1] "integer"
```

```
> v1<-c(T,FALSE)
> v2<-c(10L,20L)
> v3<-c(3.5,12.2)
> v4<-c("system","dynamics")
>
> typeof(v1)
[1] "logical"
> typeof(v2)
[1] "integer"
> typeof(v3)
[1] "double"
> typeof(v4)
[1] "character")
```



Index

- The concept of an index is powerful in R, as it allows access to individual data elements of a vector, using the square brackets notation.
- In R, unlike programming languages such as C and Java, the index for a vector starts at 1.

```
> v1  
[1] 1 4 9 16 25  
  
> v1[1]  
[1] 1  
  
> v1[2]  
[1] 4  
  
> v1[5]  
[1] 25
```



Creating sequences

- The colon operator (:) generates regular sequences within a specified range.

```
> v1<-1:10  
> v1  
[1] 1 2 3 4 5 6 7 8 9 10  
  
> v2<-3:13  
> v2  
[1] 3 4 5 6 7 8 9 10 11 12 13
```



Sequences as indices

- Sequences can be used as indices for vectors, with the minus sign used for exclusion

```
> v1  
[1] 1 4 9 16 25  
  
> v1[1:2]  
[1] 1 4  
  
> v1[-1]  
[1] 4 9 16 25  
  
> v1[(-(1:3))]  
[1] 16 25
```



Vectorization

- A powerful feature of R is that it supports *vectorization*
- Functions can operate on every element of a vector, and return the results of each individual operation in a new vector.

```
> v1  
[1] 1 2 3 4 5  
  
> r<-sqrt(v1)  
  
> r  
[1] 1.000000 1.414214 1.732051 2.000000 2.236068
```



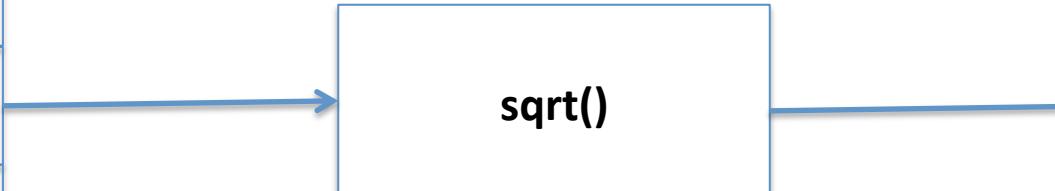
Key Idea

Input Vector

1
4
9
16
25

Output Vector

1
2
3
4
5



Conditional Expressions on Vectors

- Conditional expressions can be applied to vectors, and this operation returns a boolean vector

```
> v1  
[1] 1 4 9 16 25  
  
> v1 %% 2 == 0  
[1] FALSE TRUE FALSE TRUE FALSE
```

Operators	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	not x
x y	x OR y
x & y	x AND y

<http://www.statmethods.net/management/operators.html>



Boolean vectors used as indices

- A target the vector can be filtered by the TRUE locations in the boolean vector.

```
> v1  
[1] 1 4 9 16 25  
  
> b1<-v1 %% 2 == 1  
  
> b1  
[1] TRUE FALSE TRUE FALSE TRUE  
  
> v1[b1]  
[1] 1 9 25
```





Challenge 1



- Create an R vector of squares of 1 to 10
- Find the minimum
- Find the maximum
- Display all those greater than the mean
- Display all those less than the mean
- *Display every second vector element*
- *Find the index location of the maximum value*



(2) Functions

- A function is a group of instructions that:
 - takes input,
 - uses the input to compute other value, and
 - returns a result.
- Functions are declared:
 - using the **function** reserved word
 - are objects

```
function(arguments){  
  expression  
}
```



Example 1

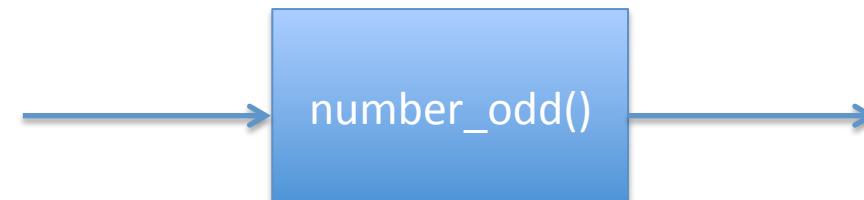


```
add<-function(a,b){  
  a+b  
}
```

```
> add(3,4)  
[1] 7  
> add(-1,1)  
[1] 0
```



Example 2



```
number_odd<-function(v){  
  length(v[v%%2 == 1])  
}
```

```
> v<-1:10  
> v  
[1]  1  2  3  4  5  6  7  8  9 10  
> ans<-number_odd(v)  
> ans  
[1] 5
```



Example 3



```
simtime<-function(start, finish, DT=1){  
  seq(start,finish,by = DT)  
}
```

```
> simtime(1,10,0.25)  
[1] 1.00 1.25 1.50 1.75 2.00 2.25 2.50 2.75 3.00 3.25 3.50 3.75 4.00  
[14] 4.25 4.50 4.75 5.00 5.25 5.50 5.75 6.00 6.25 6.50 6.75 7.00 7.25  
[27] 7.50 7.75 8.00 8.25 8.50 8.75 9.00 9.25 9.50 9.75 10.00
```





Challenge 2



- Write a function that takes in a vector and returns a vector of even numbers
- *Write a function that returns the unique values in a vector (hint: use the R function duplicated)*



(3) Lists

- A list is a vector that can contain different types
- Flexible for returning function results
- Lists can be recursive (a list can contain a list)

```
v1<-c(1,2,3)
v2<-c("One","Two","Three")
l<-list(Numbers=v1,Names=v2)
```

```
> l
$Numbers
[1] 1 2 3

$Names
[1] "One"  "Two"  "Three"
```



Indexing: [], [[]], \$

```
> str(l)
List of 2
 $ Numbers: num [1:3] 1 2 3
 $ Names : chr [1:3] "One" "Two" "Three"
```

```
> l[1]
$Numbers
[1] 1 2 3
```

```
> l[[1]]
[1] 1 2 3
```

```
> l[2]
$Names
[1] "One" "Two" "Three"
```

```
> l[[2]]
[1] "One" "Two" "Three"
```

```
> l$Numbers
[1] 1 2 3
```

```
> l$Names
[1] "One" "Two" "Three"
```



(4) Data Frames

- Can be viewed as a set of vectors, organised into a column format
- Each column can have a different data type

```
> ds<-data.frame(mpg)
> head(ds)
```

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	f1	class
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
4	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact



Filtering Data Frame using matrix notation [row,col]

```
> ds[1,]
  manufacturer model displ year cyl      trans drv cty hwy fl   class
1       audi     a4    1.8 1999    4 auto(l5)   f  18  29  p compact
> ds[1:4,]
  manufacturer model displ year cyl      trans drv cty hwy fl   class
1       audi     a4    1.8 1999    4 auto(l5)   f  18  29  p compact
2       audi     a4    1.8 1999    4 manual(m5) f  21  29  p compact
3       audi     a4    2.0 2008    4 manual(m6) f  20  31  p compact
4       audi     a4    2.0 2008    4 auto(av)   f  21  30  p compact
> ds[1:4,1:4]
  manufacturer model displ year
1       audi     a4    1.8 1999
2       audi     a4    1.8 1999
3       audi     a4    2.0 2008
4       audi     a4    2.0 2008
```



Subset data using conditionals

```
> b<-ds$manufacturer=="audi" & ds$year==1999
```

```
> ds[b,]
```

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
8	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
9	audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
12	audi	a4 quattro	2.8	1999	6	auto(l5)	4	15	25	p	compact
13	audi	a4 quattro	2.8	1999	6	manual(m5)	4	17	25	p	compact
16	audi	a6 quattro	2.8	1999	6	auto(l5)	4	15	24	p	midsize
.											



Adding new columns

```
> ds$status<-ifelse(ds$year<2005,"OLD","NEW")
> head(ds)
  manufacturer model displ year cyl      trans drv cty hwy fl class status
1       audi     a4   1.8 1999    4  auto(l5)   f  18  29  p compact    OLD
2       audi     a4   1.8 1999    4 manual(m5)   f  21  29  p compact    OLD
3       audi     a4   2.0 2008    4 manual(m6)   f  20  31  p compact   NEW
4       audi     a4   2.0 2008    4  auto(av)    f  21  30  p compact   NEW
5       audi     a4   2.8 1999    6  auto(l5)    f  16  26  p compact    OLD
6       audi     a4   2.8 1999    6 manual(m5)   f  18  26  p compact    OLD
```



subset() function

```
> s<-subset(ds,ds$displ==1.8 & ds$status=="OLD")
> s
   manufacturer      model displ year cyl      trans drv cty hwy fl class status
1          audi         a4    1.8 1999  4 auto(l5)   f  18  29  p compact  OLD
2          audi         a4    1.8 1999  4 manual(m5)  f  21  29  p compact  OLD
8          audi  a4 quattro    1.8 1999  4 manual(m5)  4  18  26  p compact  OLD
9          audi  a4 quattro    1.8 1999  4 auto(l5)   4  16  25  p compact  OLD
194         toyota    corolla    1.8 1999  4 auto(l3)   f  24  30  r compact  OLD
195         toyota    corolla    1.8 1999  4 auto(l4)   f  24  33  r compact  OLD
196         toyota    corolla    1.8 1999  4 manual(m5)  f  26  35  r compact  OLD
228        volkswagen    passat    1.8 1999  4 manual(m5)  f  21  29  p midsize  OLD
229        volkswagen    passat    1.8 1999  4 auto(l5)   f  18  29  p midsize  OLD
```



Reading from Excel (install gdata)

```
library(gdata)
sim <- read.xls("workshop/R code/01 session/SimData.xlsx",
                 stringsAsFactors=FALSE)
```

```
> head(sim)
   Time Age.0.14 Age.15.39 Age.40.64 Age.Over.65
1 2014.00 1000000 1500000 2000000 500000
2 2014.13 1004170 1500830 1997500 505625
3 2014.25 1008320 1501700 1995020 511230
4 2014.38 1012460 1502590 1992550 516816
5 2014.50 1016580 1503520 1990100 522383
6 2014.63 1020690 1504470 1987670 527930
```





Challenge 3

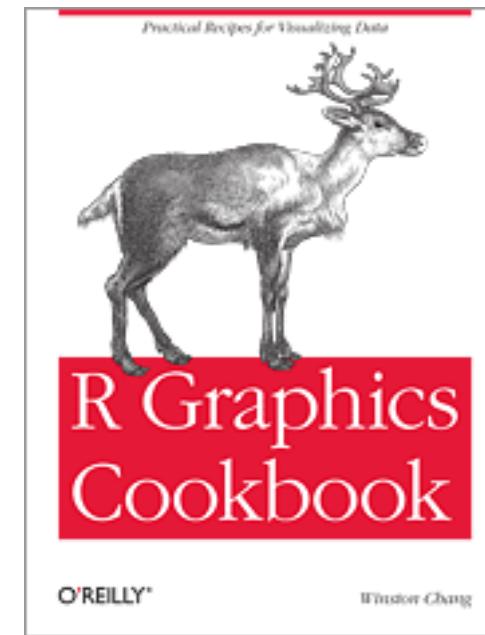


- Add the total population for year as a column to the data frame
- *Subset the data frame so that only the actual yearly values are shown (i.e. the interval between data points is 1 year)*



(5) Plotting (library ggplot2)

- Name based on Leland Wilkinson's *grammar of graphics*, which provides a formal, structured perspective on how to describe data graphics
- ggplot package developed by Hadley Wickham
- *Data must be stored in data frames*



<http://www.cookbook-r.com/Graphs/>



NUI Galway
OÉ Gaillimh

System Dynamics Modeling with R

© Jim Duggan 2016

Terminology

- The *data* is what we want to visualise. It consists of variables, which are stored as columns in the data frame. *The data should be in tidy data format.*
- *Geoms* are the geometric objects that are drawn to represent the data, such as bars, lines and points
- *Aesthetic attributes* are visual properties of geoms, such as x and y position, line colour, point shapes etc.
- *Mappings* from data values to aesthetics
- *Scales* that control the mappings from values in the data space to values in the aesthetic space.
- *Guides*, for example, tick marks and labels on an axis.



A simple example...

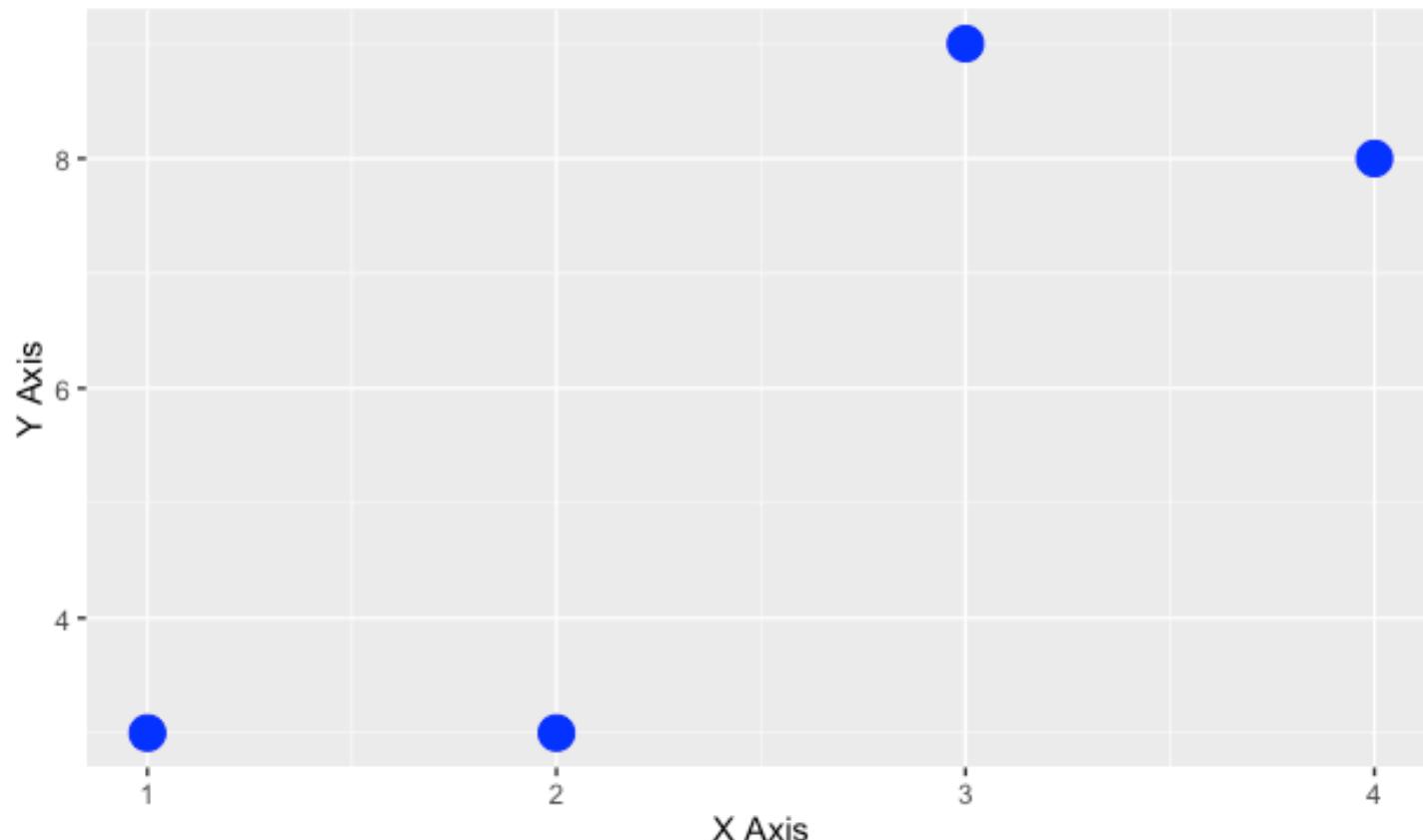
```
df<-data.frame(xval=1:4,  
                 yval=c(3,3,9,8),  
                 group=c("A","A","B","B"))
```

```
> df  
   xval yval group  
1    1    3     A  
2    2    3     A  
3    3    9     B  
4    4    8     B
```



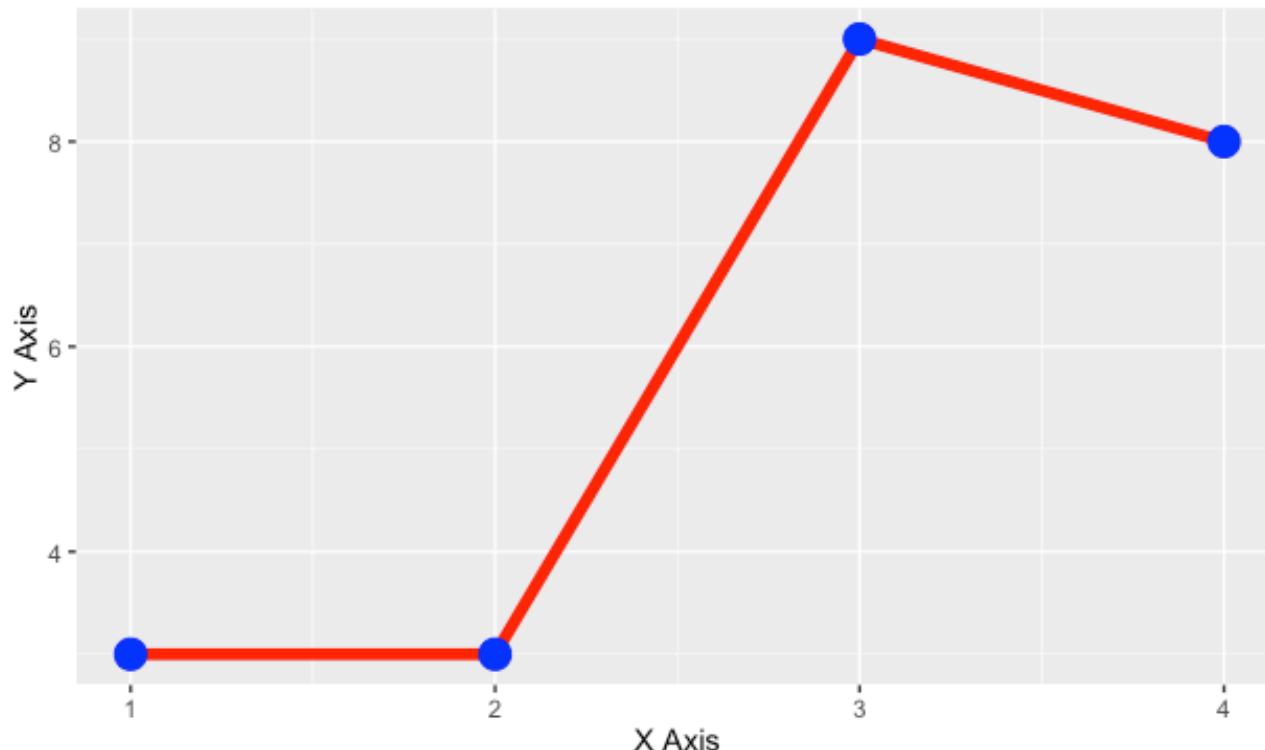
Call to ggplot()

```
ggplot(data=df,aes(x=xval,y=yval))+ geom_point(colour="blue",size=5)+  
  xlab("X Axis") + ylab("Y Axis")
```



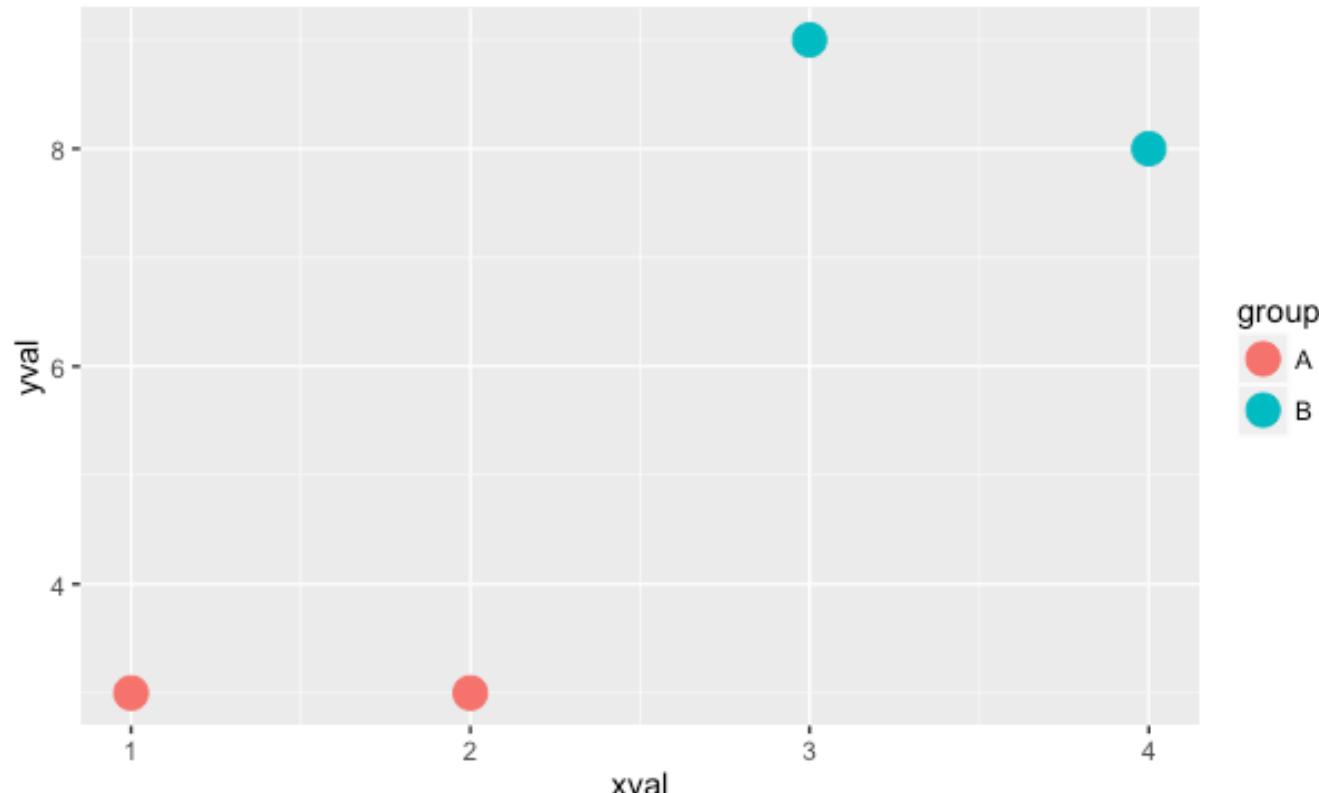
Layer different geoms...

```
ggplot(data=df,aes(x=xval,y=yval)) +  
  geom_point(colour="blue",size=5) +  
  geom_line(colour="red") + xlab("X Axis") + ylab("Y Axis")
```



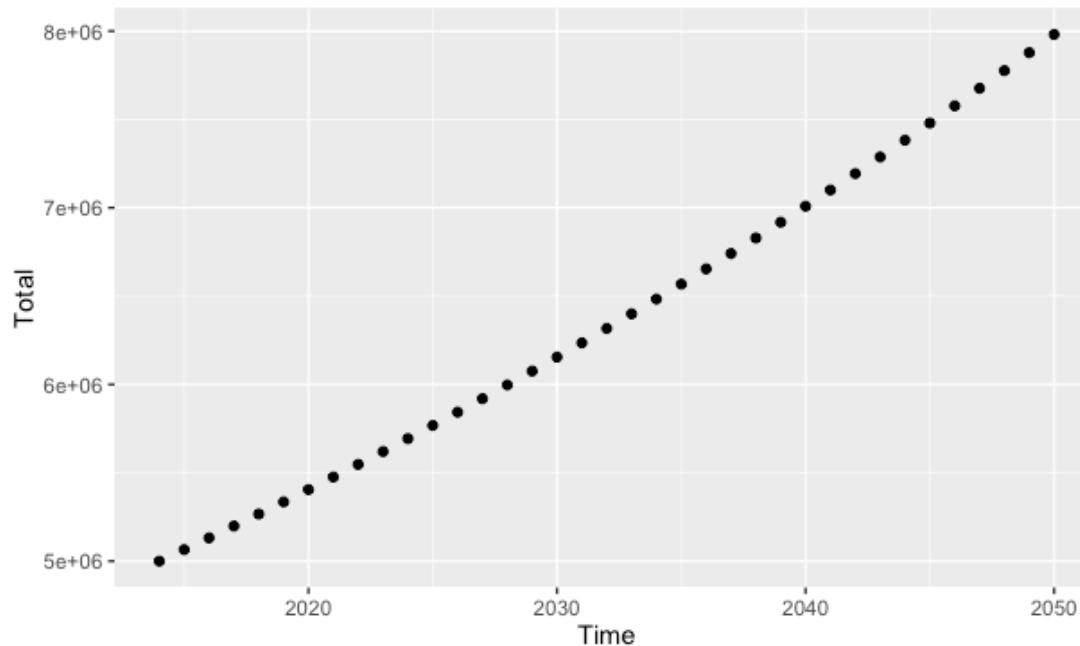
Using categorical information...

```
ggplot(df,aes(x=xval,y=yval)) + geom_point(aes(colour=group),size=5)
```



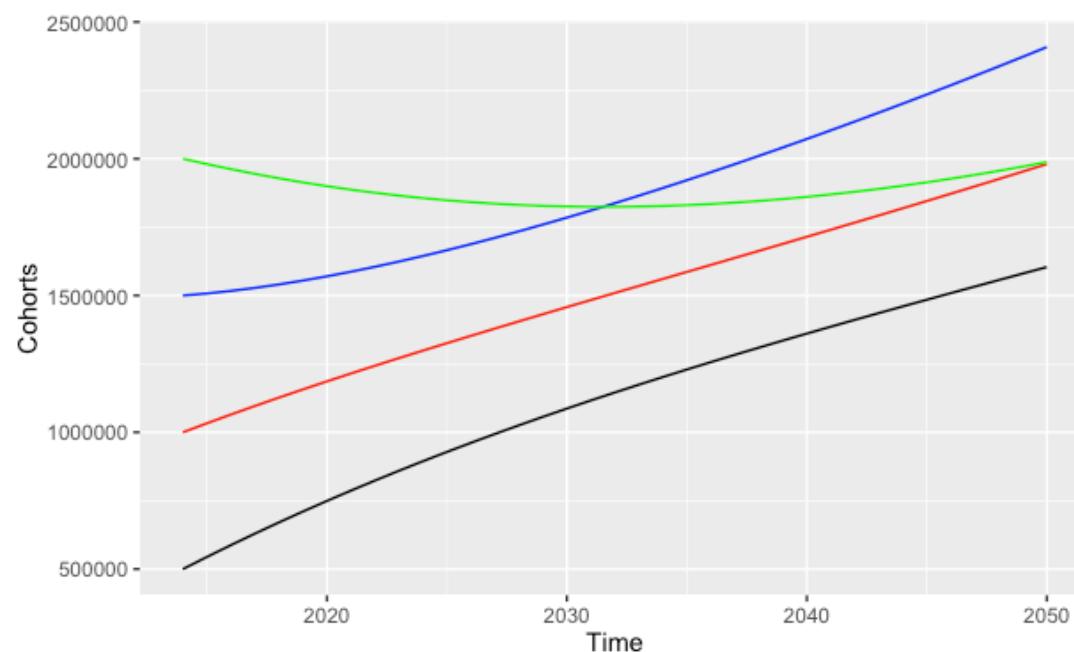
Simulation Time Series

```
sim <- read.xls("workshop/R code/01 session/SimData.xlsx",
                 stringsAsFactors=FALSE)
sim$Total<-apply(sim[,2:5],MARGIN = 1,sum)
sub<-sim[c( TRUE, rep(FALSE,7)),]
ggplot(data=sub,aes(x=Time,y=Total))+geom_point()
```



Multiple plots (slightly cumbersome)

```
ggplot(data=sub)+geom_line(aes(x=Time,y=Age.0.14),color="red")+
  geom_line(aes(x=Time,y=Age.15.39),color="blue")+
  geom_line(aes(x=Time,y=Age.40.64),color="green")+
  geom_line(aes(x=Time,y=Age.Over.65),color="black")+
  ylab("Cohorts")
```



Tidy Data

Time	Age 0-14	Age 15-39	Age 40-64	Age Over 65
2014.000	1000000.00	1500000.00	2000000.00	500000.00
2014.125	1004170.00	1500830.00	1997500.00	505625.00
2014.250	1008320.00	1501700.00	1995020.00	511230.00
2014.375	1012460.00	1502590.00	1992550.00	516816.00
2014.500	1016580.00	1503520.00	1990100.00	522383.00
2014.625	1020690.00	1504470.00	1987670.00	527930.00

- Variables in Columns
- Observations in Rows

Year	Cohort	Population
2014.000	Age 0-14	1000000.00
2014.000	Age 15-39	1500000.00
2014.000	Age 40-64	2000000.00
2014.000	Age Over 65	500000.00



`melt()` function – reshape library

`melt(data, id.vars, measure.vars)`

- **data** Data set to melt
- **id.vars** Id variables. If blank, will use all non measure.vars variables. Can be integer (variable position) or string (variable name)
- **measure.vars** Measured variables. If blank, will use all non id.vars variables. Can be integer (variable position) or string (variable name)



Population Example

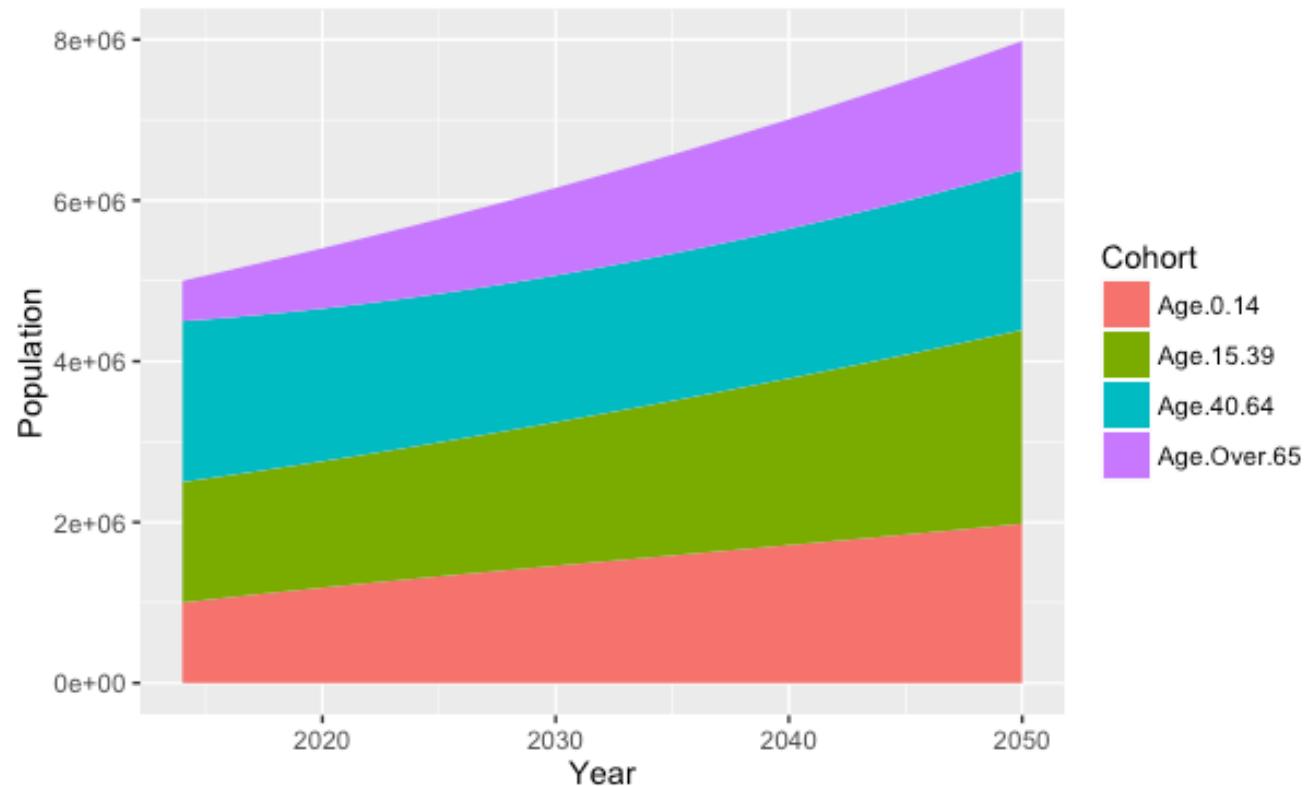
```
sub<-sim[c( TRUE, rep(FALSE,7)),]  
sub$Total<-NULL  
msub<-melt(sub,id.vars = "Time")  
names(msub)<-c("Year","Cohort","Population")
```

```
> head(msub)  
  Year Cohort Population  
1 2014 Age.0.14 1000000  
2 2015 Age.0.14 1032940  
3 2016 Age.0.14 1065020  
4 2017 Age.0.14 1096320  
5 2018 Age.0.14 1126900  
6 2019 Age.0.14 1156830
```



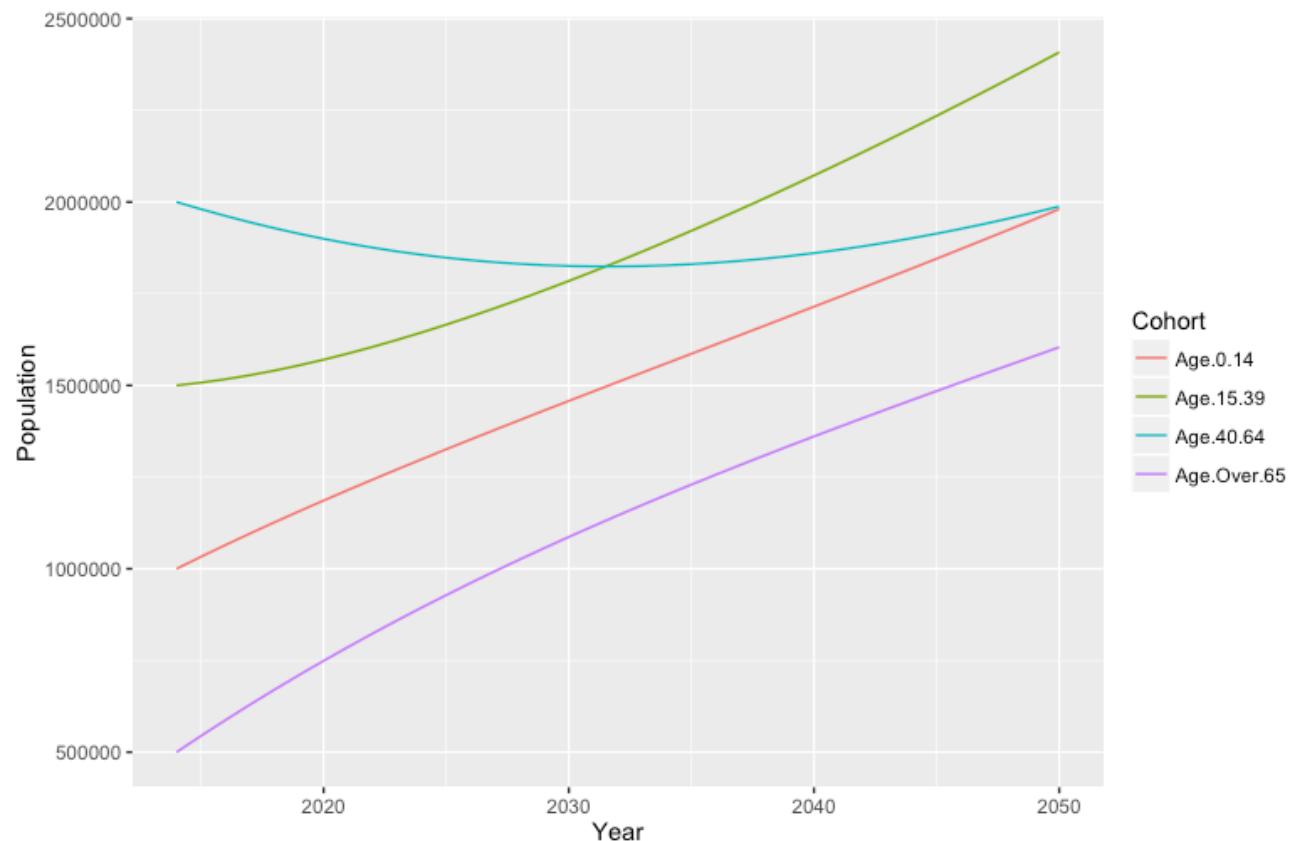
ggplot “likes” tidy data

```
ggplot(data=msub) + geom_area(aes(x=Year,y=Population,fill=Cohort))
```



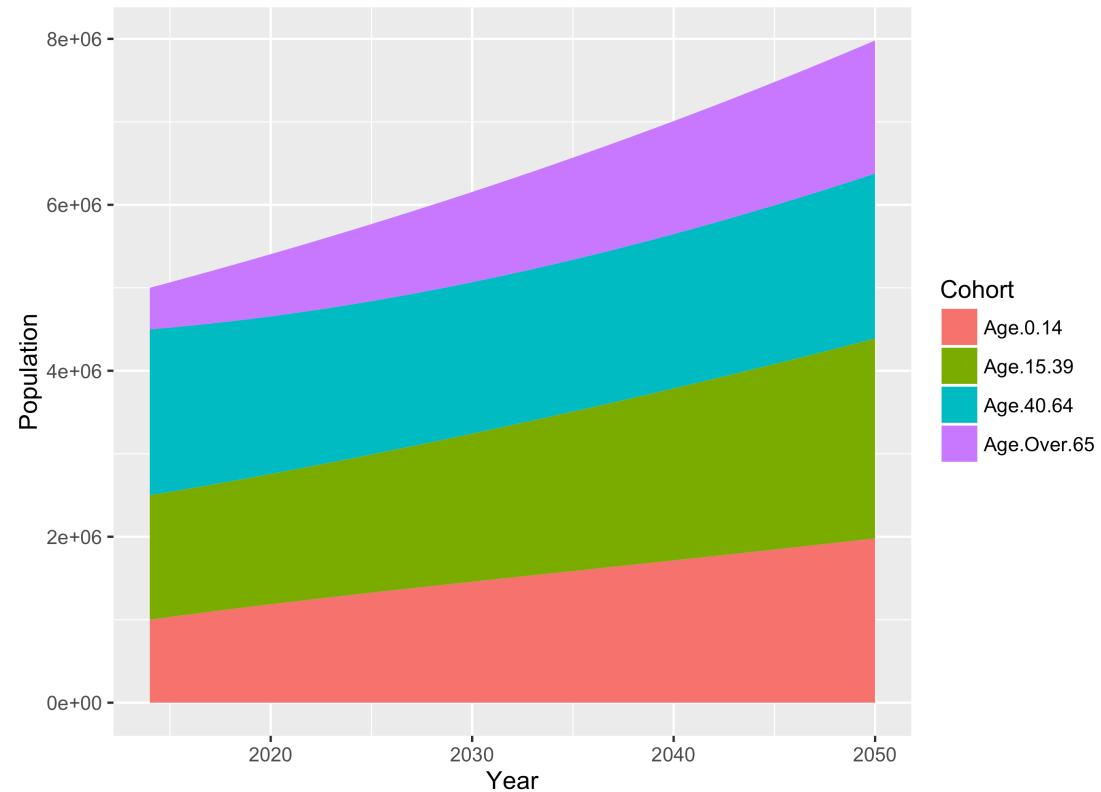
Another example...

```
ggplot(data=msub) + geom_line(aes(x=Year,y=Population,colour=Cohort))
```



Controlling Plot Resolution ggsave()

```
p1<-ggplot(data=msub) + geom_area(aes(x=Year,y=Population,fill=Cohort))  
ggsave(file="workshop/models/01 session/hq_plot.png", height=5, width=7,  
       units="in",dpi=400,p1)
```



Other plots (mpg - ggplot2 H. Wickham)

A data frame with 234 rows and 11 variables

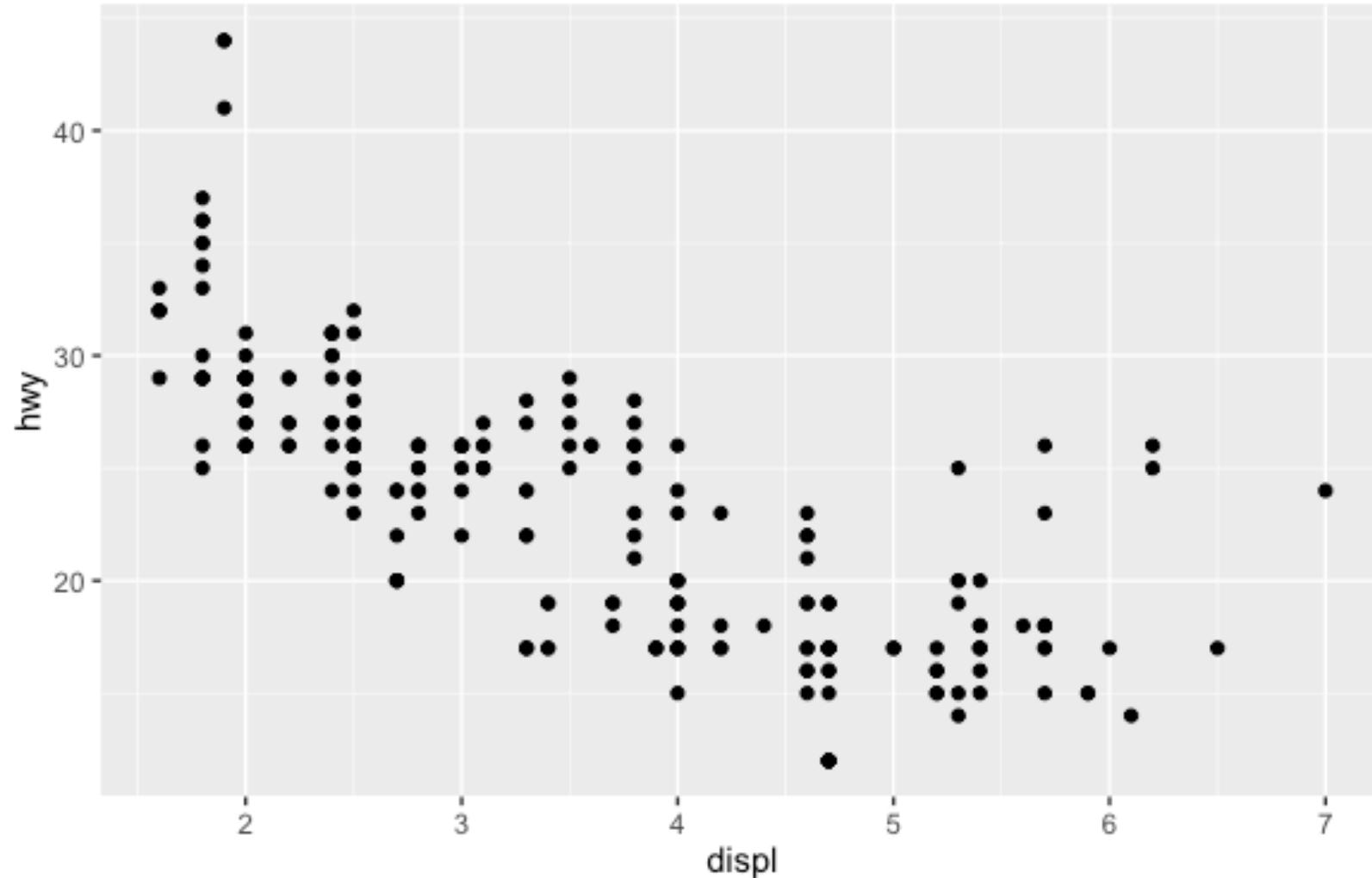
- manufacturer.
- model.
- displ. engine displacement, in litres
- year.
- cyl. number of cylinders
- trans. type of transmission
- drv. f = front-wheel drive, r = rear wheel drive, 4 = 4wd
- cty. city miles per gallon
- hwy. highway miles per gallon
- fl.
- class.

```
> head(mpg)
```

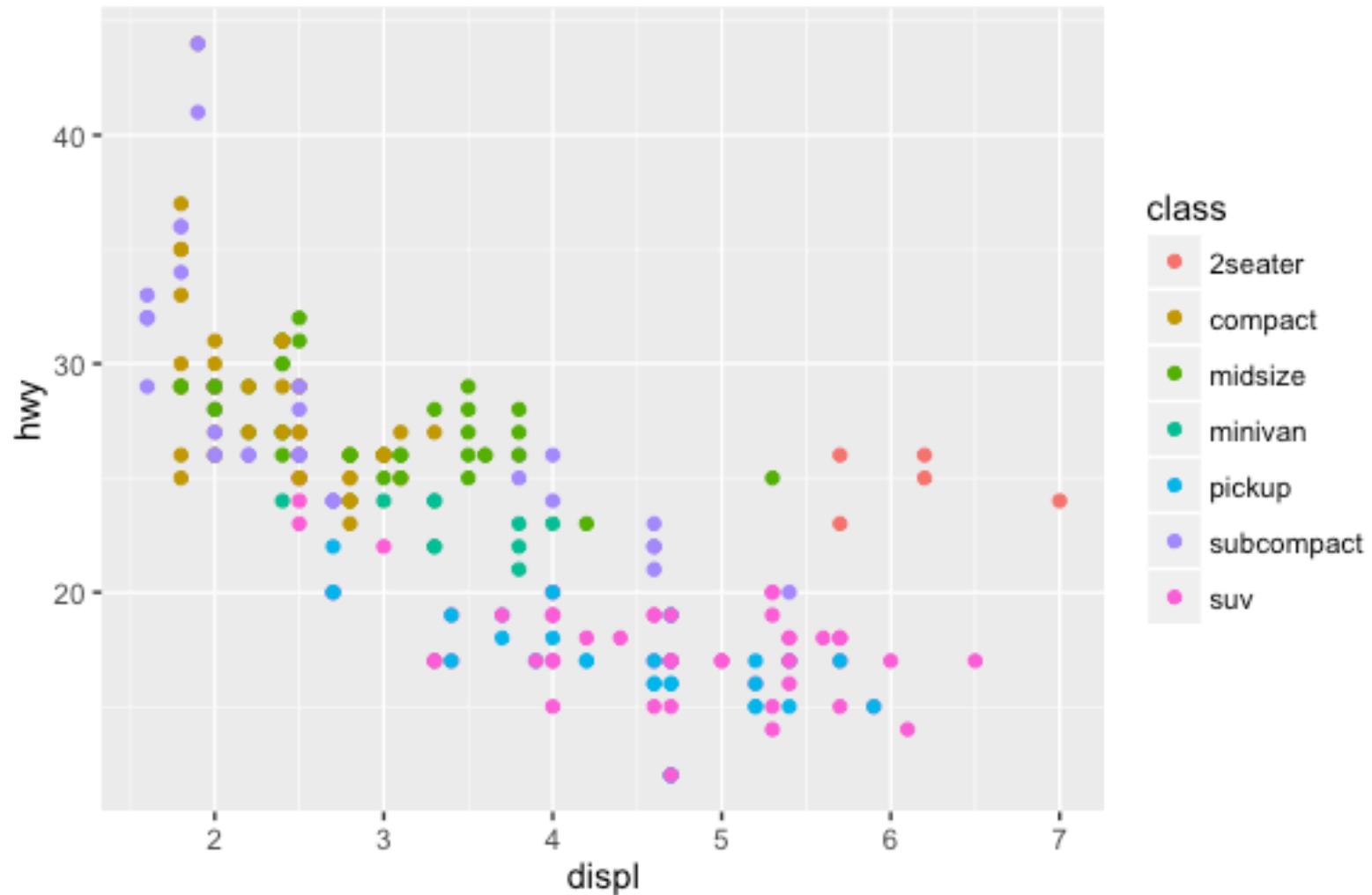
	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
4	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact



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



```
ggplot(mpg, aes(x=displ,y=hwy, colour=class)) +  
  geom_point()
```





Challenge 4



- Plot the population data on an area chart showing the percentages for each cohort over time.

