workshop05

February 3, 2019

```
In [4]: library(tidyverse)
        library(zoo)
        options(repr.plot.width=7, repr.plot.height=5)
```

0.1 What is Tidy Data or first normal form?

So far we haven't had to manipulate or clean or tidy our data at all - it has all been in a ready to use format. Tidy data is where:

- Each observation is a row
- Each variable (attribute) is a column
- Each Individual cell is a value
- Data of the same kind is stored in one table. If multiple tables are used, they must contain a column to link them.


```
In [2]: tidy <- read.table(header=TRUE, text='
          COUNTRY VALUE SALES
          AUS   Revenue     465
          ENG   Revenue     795
          CHN   Revenue     600
          AUS   Costs     454
          ENG   Costs     Missing
          CHN   Costs     700 ')
        tidy <- data.frame(tidy)
        print(tidy,row.names=TRUE)
```

```
  COUNTRY   VALUE   SALES
1     AUS Revenue     465
2     ENG Revenue     795
3     CHN Revenue     600
4     AUS   Costs     454
5     ENG   Costs Missing
6     CHN   Costs     700
```

Tidy datasets are easy to manipulate, model and visualise.

0.2 Why are we learning how to clean and tidy data?

It is said that approximately **80%** of the time spent doing analysis is taken up by tidying and cleaning up data.

Furthermore, it is directly relevant to you! For many graduate jobs and internships, you will be dealing with data of some sort in one way or another, whether that be collecting, analysing or presenting. Being familiar with different data structures and tidying means:

- You can do things with the data that your managers can't
- You can get your job done quicker, and more accurately

0.3 Why is data not tidy?

- Poor collection procedures
- Data structure set-up for a specific purpose
- Data comes from different sources

```
In [3]: messy <- read.table(header=FALSE, text='
          Country:AUS   465     454
          Country:ENG   795     Missing
          Country:CHN   600     700
                ')
        messy <- data.frame(messy)
        print(messy,row.names=FALSE)
```

```
          V1  V2      V3
 Country:AUS 465     454
 Country:ENG 795 Missing
 Country:CHN 600     700
```


0.4 How to make data tidy?

Process of cleaning data and preparing it for the analysis includes:

- Viewing the structure of the data, identifying outliers, obvious errors, missing values. Commonly used functions: `class`, `dim`, `names`, `str`, `summary` and others.
- Data manipulation (often with strings and dates) using *lubridate*, *stringr* and other packages.
- Conversion to an appropriate format: long/wide format (will be discussed below), split of one table into multiple tables with links.

0.5 What are the main forms and structures of data?

- With data, we have **Keys** and **Values**. These come in pairs. For every value there is a key!
- For example, Name: Luke and Name: Jill are key-value pairs. Name is the key in both cases, while Luke and Jill are the values.
- Each cell in a spreadsheet is a value in a key-value pair.
- In tidy data, each column name would be a key, and each cell within the column would be a value

0.5.1 Wide form

Each observation/individual from our sample is in one row only. Each row consists of many measurements for the same observation/individual, and the columns store the value for one of these measurements. E.g. for the protein set, we have 25 as sample size, 25 observations and 25 rows of data. We have 2 columns to identify the observations, and 9 columns to store the measurements.

```
In [4]: protein.df <- read.csv("protein.csv")
        head(protein.df)
```

Country	Location	RedMeat	WhiteMeat	Eggs	Milk	Fish	Cereals	Starch	Nuts	Fr.Veg
Albania	E	10.1	1.4	0.5	8.9	0.2	42.3	0.6	5.5	1.7
Austria	W	8.9	14.0	4.3	19.9	2.1	28.0	3.6	1.3	4.3
Belgium	W	13.5	9.3	4.1	17.5	4.5	26.6	5.7	2.1	4.0
Bulgaria	E	7.8	6.0	1.6	8.3	1.2	56.7	1.1	3.7	4.2
Czechoslovakia	E	9.7	11.4	2.8	12.5	2.0	34.3	5.0	1.1	4.0
Denmark	S	10.6	10.8	3.7	25.0	9.9	21.9	4.8	0.7	2.4

0.5.2 Long form

In long form, each observation have multiple rows since each row represents one particular measurement from one particular observation/individual. Previously in the wide form there were 25 rows and 11 columns, now there are 225 rows and 4 columns! The 225 come from each observation having 9 "measurements", and each row constituting just one of these measurements: 9*25=225. Our number of columns has decreased from 11 to 4. We have eliminated the 9 previous columns that contained our measurements but created 2 more columns - one to store a value, and one to store the "key" for that value (the key was originally a column name, and the value was originally a cell within the column). 11-9+2=4


```
In [5]: protein.long <- gather(protein.df, Food, Value, -Country, -Location)
        protein.long[1:18,]
```

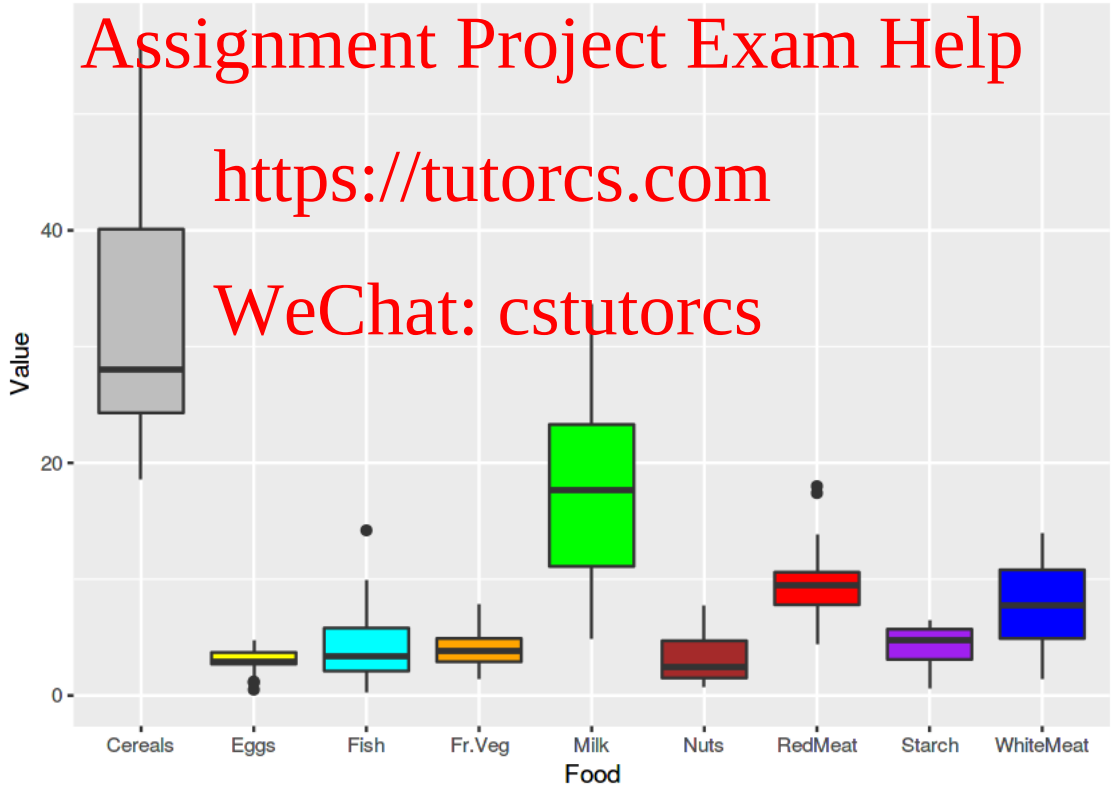
Country	Location	Food	Value
Albania	E	RedMeat	10.1
Austria	W	RedMeat	8.9
Belgium	W	RedMeat	13.5
Bulgaria	E	RedMeat	7.8
Czechoslovakia	E	RedMeat	9.7
Denmark	S	RedMeat	10.6
E Germany	E	RedMeat	8.4
Finland	S	RedMeat	9.5
France	W	RedMeat	18.0
Greece	W	RedMeat	10.2
Hungary	E	RedMeat	5.3
Ireland	W	RedMeat	13.9
Italy	W	RedMeat	9.0
Netherlands	W	RedMeat	9.5
Norway	S	RedMeat	9.4
Poland	E	RedMeat	6.9
Portugal	W	RedMeat	6.2
Romania	E	RedMeat	6.2

0.5.3 Wide or long data?

Clean data should be either wide or long. What to choose? In general, wide format is more readable and long format is easier to analyse. Let's take a look at the example.

We want to use ggplot2 to make some box-plots of the protein data. We could use this code and the wide form original data:

```
In [6]: w <- ggplot(data=protein.df)
        w+geom_boxplot(aes(x="RedMeat", y=RedMeat), fill="red")+
          geom_boxplot(aes(x="WhiteMeat", y=WhiteMeat), fill="blue")+
          geom_boxplot(aes(x="Milk", y=Milk), fill="green")+
          geom_boxplot(aes(x="Fish", y=Fish), fill="cyan")+
          geom_boxplot(aes(x="Eggs", y=Eggs), fill="yellow")+
          geom_boxplot(aes(x="Starch", y=Starch), fill="purple")+
          geom_boxplot(aes(x="Fr.Veg", y=Fr.Veg), fill="orange")+
          geom_boxplot(aes(x="Nuts", y=Nuts), fill="brown")+
          geom_boxplot(aes(x="Cereals", y=Cereals), fill="grey")+
          xlab("Food")+ylab("Value")
```


```
In [7]: #Now let's use the long form of the data...

        p <- ggplot(data=protein.long, aes(x=Food, y=Value))
        p+geom_boxplot(aes(fill=Food))+theme(legend.position="none")
```

0.6 That's amazing! How to switch between wide and long formats?!

We use a package called *tidyr*.

- The function `gather` takes us from wide to long
- The function `spread` takes us from long to wide
- We only look at `gather` for now, which is structured below:

```
data.long <- gather(data=..., key=..., value=..., columns)
```

- data is our data frame in wide form
- key is what you want to name the new column that will contain the original column names/keys from your wide data
- value is what you want to name the new column you want your values to be stored in
- columns is where you enter the names for the columns from the original wide data that you want to "collapse".

The arguments **key** and **value** are the columns that you create in your new long data frame, whereas the argument **columns** is what you "destroy" from your original wide data, the remaining of which get put into **key** and **value**.

0.7 How do we apply this to the protein data?

- data: our data frame in wide form is **protein**
- key: Let's call our key **Food** - this will contain the names for the types of food - that is, where the wide column names will go.
- value - our values refer to percentages of dietary protein from the food , so let's call our value column **Percentage**.
- columns - this is where we input the columns from our wide data set. So we need to put in **RedMeat, WhiteMeat, Eggs, Milk, Fish, Cereals, Starch, Nuts, Fr.Veg**

```
In [13]: protein.df <- read.csv("protein.csv")
         protein.long <- gather(data=protein.df, key=Food, value=Percentage, RedMeat,
                                WhiteMeat, Eggs, Milk, Fish,
                                Cereals, Starch, Nuts, Fr.Veg)
```

Let's now compare the data frames:

```
In [12]: #__Wide:__
         protein.df[1:7,]
```



```
         #__Long:__
         protein.long[1:12,]
```


Country	Location	RedMeat	WhiteMeat	Eggs	Milk	Fish	Cereals	Starch	Nuts	Fr.Veg
Albania	E	10.1	1.4	0.5	8.9	0.2	42.3	0.6	5.5	1.7
Austria	W	8.9	14.0	4.3	19.9	2.1	28.0	3.6	1.3	4.3
Belgium	W	13.5	9.3	4.1	17.5	4.5	26.6	5.7	2.1	4.0
Bulgaria	E	7.8	6.0	1.6	8.3	1.2	56.7	1.1	3.7	4.2
Czechoslovakia	E	9.7	11.4	2.8	12.5	2.0	34.3	5.0	1.1	4.0
Denmark	S	10.6	10.8	3.7	25.0	9.9	21.9	4.8	0.7	2.4
E Germany	E	8.4	11.6	3.7	11.1	5.4	24.6	6.5	0.8	3.6

Country	Location	Food	Percentage
Albania	E	RedMeat	10.1
Austria	W	RedMeat	8.9
Belgium	W	RedMeat	13.5
Bulgaria	E	RedMeat	7.8
Czechoslovakia	E	RedMeat	9.7
Denmark	S	RedMeat	10.6
E Germany	E	RedMeat	8.4
Finland	S	RedMeat	9.5
France	W	RedMeat	18.0
Greece	W	RedMeat	10.2
Hungary	E	RedMeat	5.3
Ireland	W	RedMeat	13.9

So what has happened?

- We started with a data frame in wide form called "protein"
- We have created 2 columns, one named "Food" to store what used to be our keys/column names; and one named "Percentage" to store our values

- We have taken the information that used to be stored in the columns called RedMeat, WhiteMeat, etc. and put it into the new columns we created.

0.8 Resources

- http://garrettgman.github.io/tidying/ - a very thorough explanation of how to tidy data if you want to learn more
- https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html - five most common problems with messy datasets, along with their remedies
- The documentation for `tidyr` - https://cran.r-project.org/web/packages/tidyr/tidyr.pdf

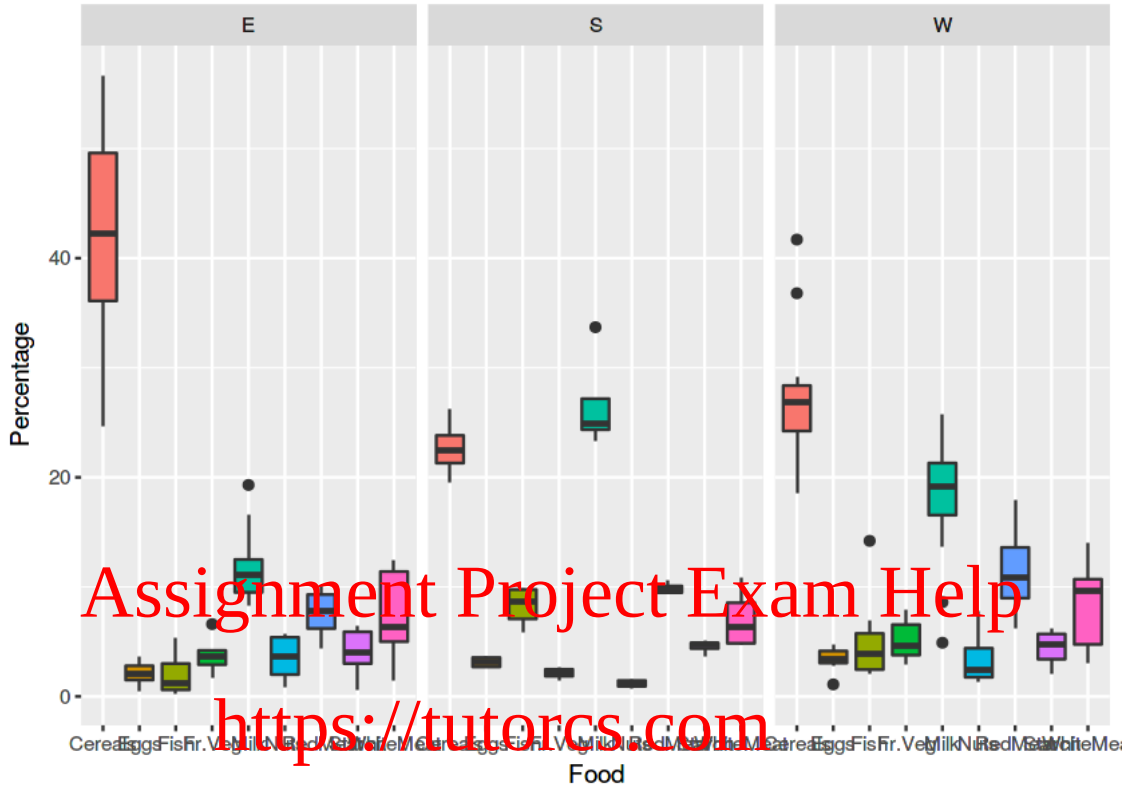
0.8.1 Exercise 1

Let's now make use of the long format by creating even more complicated plots. Previously, we created one boxplot across the whole data-set. If we want to create a box plot across all food types by region, we usually sub-set the data. To do this all at once, there is a new command called `facet_wrap(~VARIABLE)` that allows you to do this in one go. Create a boxplot as before but append `facet_wrap(~Location)`:

```
In [15]: protein_long
```

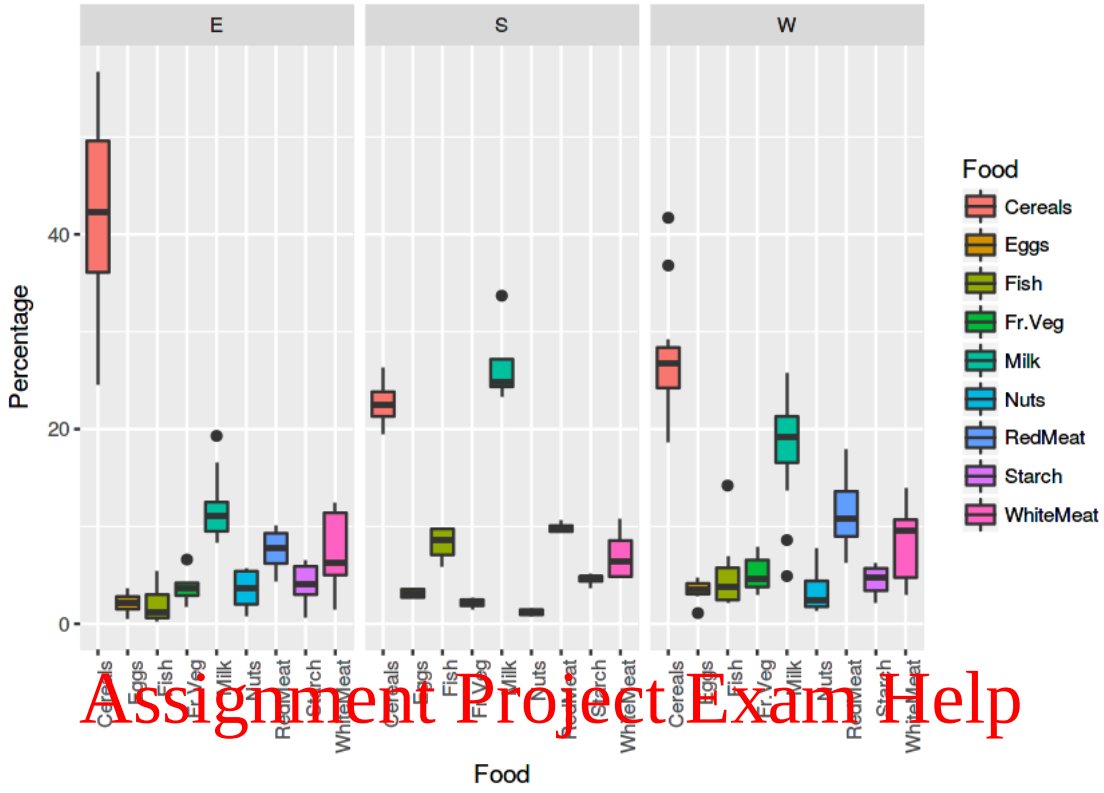
Country	Location	Food	Percentage
Albania	E	RedMeat	10.1
Austria	W	RedMeat	8.9
Belgium	W	RedMeat	13.5
Bulgaria	E	RedMeat	7.8
Czechoslovakia	E	RedMeat	9.7
Denmark	S	RedMeat	10.6
E Germany	E	RedMeat	8.4
Finland	S	RedMeat	9.5
France	W	RedMeat	18.0
Greece	W	RedMeat	10.2
Hungary	E	RedMeat	5.3
Ireland	W	RedMeat	13.9
Italy	W	RedMeat	9.0
Netherlands	W	RedMeat	9.5
Norway	S	RedMeat	9.4
Poland	E	RedMeat	6.9
Portugal	W	RedMeat	6.2
Romania	E	RedMeat	6.2
Spain	W	RedMeat	7.1
Sweden	S	RedMeat	9.9
Switzerland	W	RedMeat	13.1
UK	W	RedMeat	17.4
USSR	E	RedMeat	9.3
W Germany	W	RedMeat	11.4
Yugoslavia	E	RedMeat	4.4
Albania	E	WhiteMeat	1.4
Austria	W	WhiteMeat	11.0
Belgium	W	WhiteMeat	9.3
Bulgaria	E	WhiteMeat	6.0
Czechoslovakia	E	WhiteMeat	11.4
Switzerland	W	Nuts	2.4
UK	W	Nuts	3.4
USSR	E	Nuts	3.4
W Germany	W	Nuts	1.5
Yugoslavia	E	Nuts	5.7
Albania	E	Fr.Veg	1.7
Austria	W	Fr.Veg	4.3
Belgium	W	Fr.Veg	4.0
Bulgaria	E	Fr.Veg	4.2
Czechoslovakia	E	Fr.Veg	4.0
Denmark	S	Fr.Veg	2.4
E Germany	E	Fr.Veg	3.6
Finland	S	Fr.Veg	1.4
France	W	Fr.Veg	6.5
Greece	W	Fr.Veg	6.5
Hungary	E	Fr.Veg	4.2
Ireland	W	Fr.Veg	2.9
Italy	W	Fr.Veg	6.7
Netherlands	W	Fr.Veg	3.7
Norway	S	Fr.Veg	2.7
Poland	E	Fr.Veg	6.6

```
In [16]: p <- ggplot(data=protein.long, aes(x=Food, y=Percentage))
         p+geom_boxplot(aes(fill=Food))+facet_wrap(~Location)+theme(legend.position="none")
```



You will notice the legend is not readable. To fix this we need to rotate the axis text. Append `theme(axis.text.x = element_text(angle = 90, hjust = 1))`:

```
In [17]: p <- ggplot(data=protein.long, aes(x=Food, y=Percentage))
         p+geom_boxplot(aes(fill=Food))+facet_wrap(~Location)+theme(axis.text.x = element_text(a
```



What do you see? Discuss it with your table.

0.8.2 Exercise 2 (difficult): Learn how to replicate graphs

Let's practice our skills on a new data set. This is now day 5, let's practice all the skill we have learned so far. Read in the file `landdata-states.csv`. You will notice that the data-set covers housing prices in the US over time. You need to re-arrange the data in a way that you can run the plot command at the end of this exercise.

```
In [12]: housing.df <-read.table(file = "landdata-states.csv",
                                 header = TRUE,
                                 sep = ",")
```

Hint: To convert the string `Date` into a numeric value this command will help you in your transformation, what does it do ?

```
In [10]: housing1.df <-
         housing.df %>%
             mutate( prepeared_date = paste(substr(Date,1,4),'-',substr(Date,5,5)),
                     Time           = as.yearqtr(paste(substr(Date,1,4),'-',substr(Date,5,5)), f
         housing1.df %>%
         head()
```

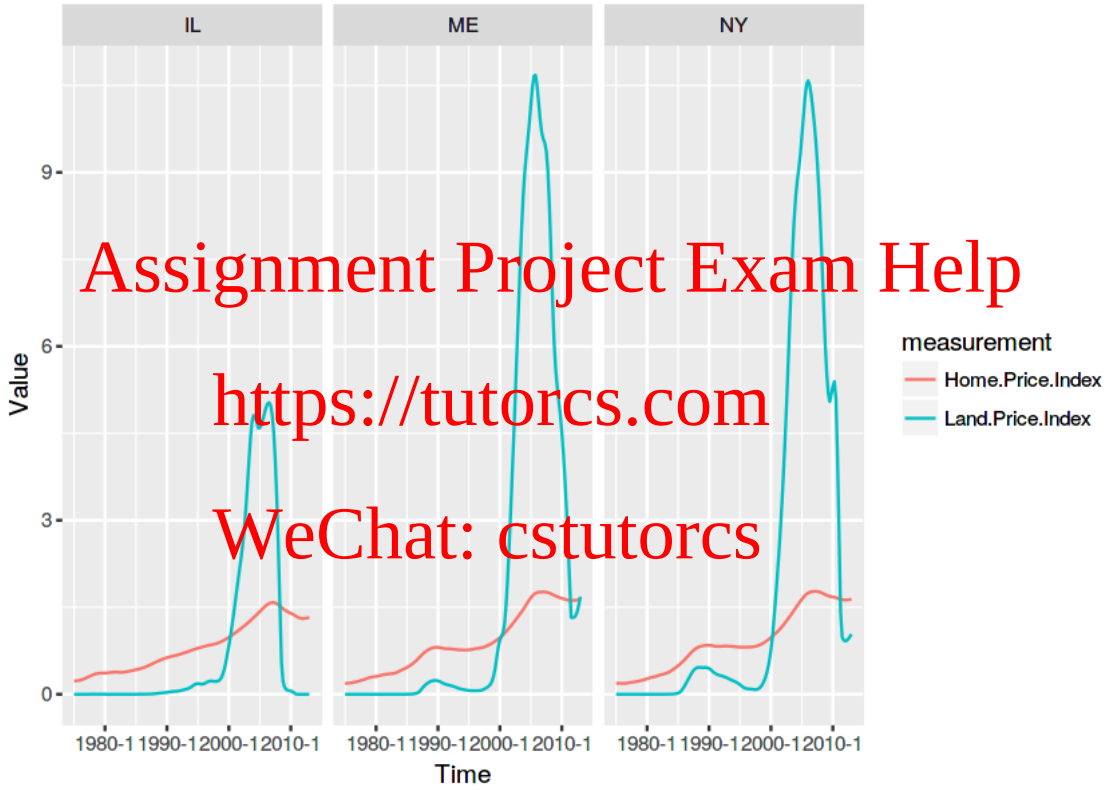
State	region	Date	Home.Value	Structure.Cost	Land.Value	Land.Share..Pct.	Home.Price.Index
AK	West	20101	224952	160599	64352	28.6	1.481
AK	West	20102	225511	160252	65259	28.9	1.484
AK	West	20093	225820	163791	62029	27.5	1.486
AK	West	20094	224994	161787	63207	28.1	1.481
AK	West	20074	234590	155400	79190	33.8	1.544
AK	West	20081	233714	157458	76256	32.6	1.538

```
In [14]: housingLong.df <-
             housing.df %>%
             mutate( prepeared_date = paste(substr(Date,1,4),'-',substr(Date,5,5)),
                     Time           = as.yearqtr(paste(substr(Date,1,4),'-',substr(Date,5,5)),

             gather(data=., key=measurement, value=Value, Home.Value,
                              Structure.Cost, Land.Value, Land.Share..Pct., Home.Price.Index,
                              Land.Price.Index)
         housingLong.df
```

State	region	Date	prepeared_date	Time	measurement	Value
AK	West	20101	2010 - 1	2010 Q1	Home.Value	224952
AK	West	20102	2010 - 2	2010 Q2	Home.Value	225511
AK	West	20093	2009 - 3	2009 Q3	Home.Value	225820
AK	West	20094	2009 - 4	2009 Q4	Home.Value	224994
AK	West	20074	2007 - 4	2007 Q4	Home.Value	234590
AK	West	20081	2008 - 1	2008 Q1	Home.Value	233714
AK	West	20082	2008 - 2	2008 Q2	Home.Value	232999
AK	West	20083	2008 - 3	2008 Q3	Home.Value	232164
AK	West	20084	2008 - 4	2008 Q4	Home.Value	231039
AK	West	20091	2009 - 1	2009 Q1	Home.Value	229395
AK	West	20092	2009 - 2	2009 Q2	Home.Value	227421
AK	West	19852	1985 - 2	1985 Q2	Home.Value	140207
AK	West	19853	1985 - 3	1985 Q3	Home.Value	139244
AK	West	19854	1985 - 4	1985 Q4	Home.Value	138153
AK	West	19861	1986 - 1	1986 Q1	Home.Value	136885
AK	West	19862	1986 - 2	1986 Q2	Home.Value	135164
AK	West	19863	1986 - 3	1986 Q3	Home.Value	132599
AK	West	19864	1986 - 4	1986 Q4	Home.Value	129190
AK	West	19871	1987 - 1	1987 Q1	Home.Value	125163
AK	West	19872	1987 - 2	1987 Q2	Home.Value	121016
AK	West	19873	1987 - 3	1987 Q3	Home.Value	117600
AK	West	19874	1987 - 4	1987 Q4	Home.Value	115738
AK	West	19881	1988 - 1	1988 Q1	Home.Value	115764
AK	West	19882	1988 - 2	1988 Q2	Home.Value	116617
AK	West	19883	1988 - 3	1988 Q3	Home.Value	117388
AK	West	19884	1988 - 4	1988 Q4	Home.Value	116848
AK	West	19891	1989 - 1	1989 Q1	Home.Value	114426
AK	West	19892	1989 - 2	1989 Q2	Home.Value	110811
AK	West	19893	1989 - 3	1989 Q3	Home.Value	107853
AK	West	20103	2010 - 3	2010 Q3	Home.Value	226294
DC	NA	20063	2006 - 3	2006 Q3	Land.Price.Index	3.136
DC	NA	20064	2006 - 4	2006 Q4	Land.Price.Index	3.161
DC	NA	20071	2007 - 1	2007 Q1	Land.Price.Index	3.175
DC	NA	20072	2007 - 2	2007 Q2	Land.Price.Index	3.169
DC	NA	20073	2007 - 3	2007 Q3	Land.Price.Index	3.140
DC	NA	20074	2007 - 4	2007 Q4	Land.Price.Index	3.089
DC	NA	20081	2008 - 1	2008 Q1	Land.Price.Index	3.023
DC	NA	20082	2008 - 2	2008 Q2	Land.Price.Index	2.951
DC	NA	20052	2005 - 2	2005 Q2	Land.Price.Index	2.832
DC	NA	20053	2005 - 3	2005 Q3	Land.Price.Index	2.938
DC	NA	20054	2005 - 4	2005 Q4	Land.Price.Index	3.013
DC	NA	20061	2006 - 1	2006 Q1	Land.Price.Index	3.065
DC	NA	20062	2006 - 2	2006 Q2	Land.Price.Index	3.104
DC	NA	20094	2009 - 4	2009 Q4	Land.Price.Index	2.844
DC	NA	20101	2010 - 1	2010 Q1	Land.Price.Index	2.867
DC	NA	20102	2010 - 2	2010 Q2	Land.Price.Index	2.877
DC	NA	20103	2010 - 3	2010 Q3	Land.Price.Index	2.885
DC	NA	20104	2010 - 4	2010 Q4	Land.Price.Index	2.886
DC	NA	20111	2011 - 1	2011 Q1	Land.Price.Index	2.894
DC	NA	20112	2011 - 2	2011 Q2	Land.Price.Index	2.916
DC	NA	20113	2011 - 3	2011 Q3	Land.Price.Index	2.957

```
In [15]: # GOAL: This should run, re-arange the data that this command works.
         housingLong.df %>%
             filter(State %in% c('IL','NY', 'ME'))%>%
             filter(measurement %in% c('Home.Price.Index','Land.Price.Index')) %>%
             ggplot(data=.,
                aes(x      = Time,
                    y      = Value,
                    colour = measurement)) +
             scale_x_yearqtr() +
             geom_line() +
             facet_wrap(~State)
```



0.8.3 Exercise 3

For the rest of the class, re-visit all plotting commands you have learnd so far and create visuilisations. Is there a time period you would like to zoom in ?