

# CT474: Smart Grid

## Lecture 3: Analysing Energy Data

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

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

[https://twitter.com/\\_jimduggan](https://twitter.com/_jimduggan)



# Transmission System

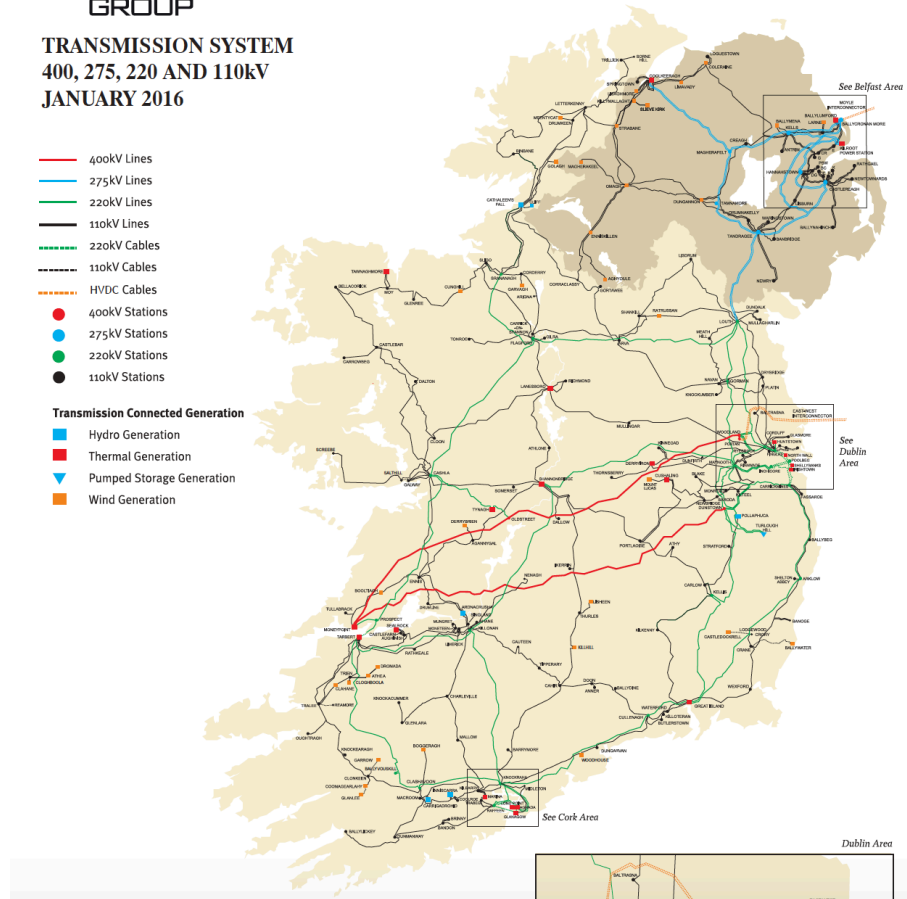


TRANSMISSION SYSTEM  
400, 275, 220 AND 110kV  
JANUARY 2016

- 400kV Lines
- 275kV Lines
- 220kV Lines
- 110kV Lines
- - - 220kV Cables
- - - 110kV Cables
- - - HVDC Cables
- 400kV Stations
- 275kV Stations
- 220kV Stations
- 110kV Stations

#### Transmission Connected Generation

- Hydro Generation
- Thermal Generation
- ▼ Pumped Storage Generation
- Wind Generation



- 400kV Lines
- 275kV Lines
- 220kV Lines
- 110kV Lines
- - - 220kV Cables
- - - 110kV Cables
- - - HVDC Cables

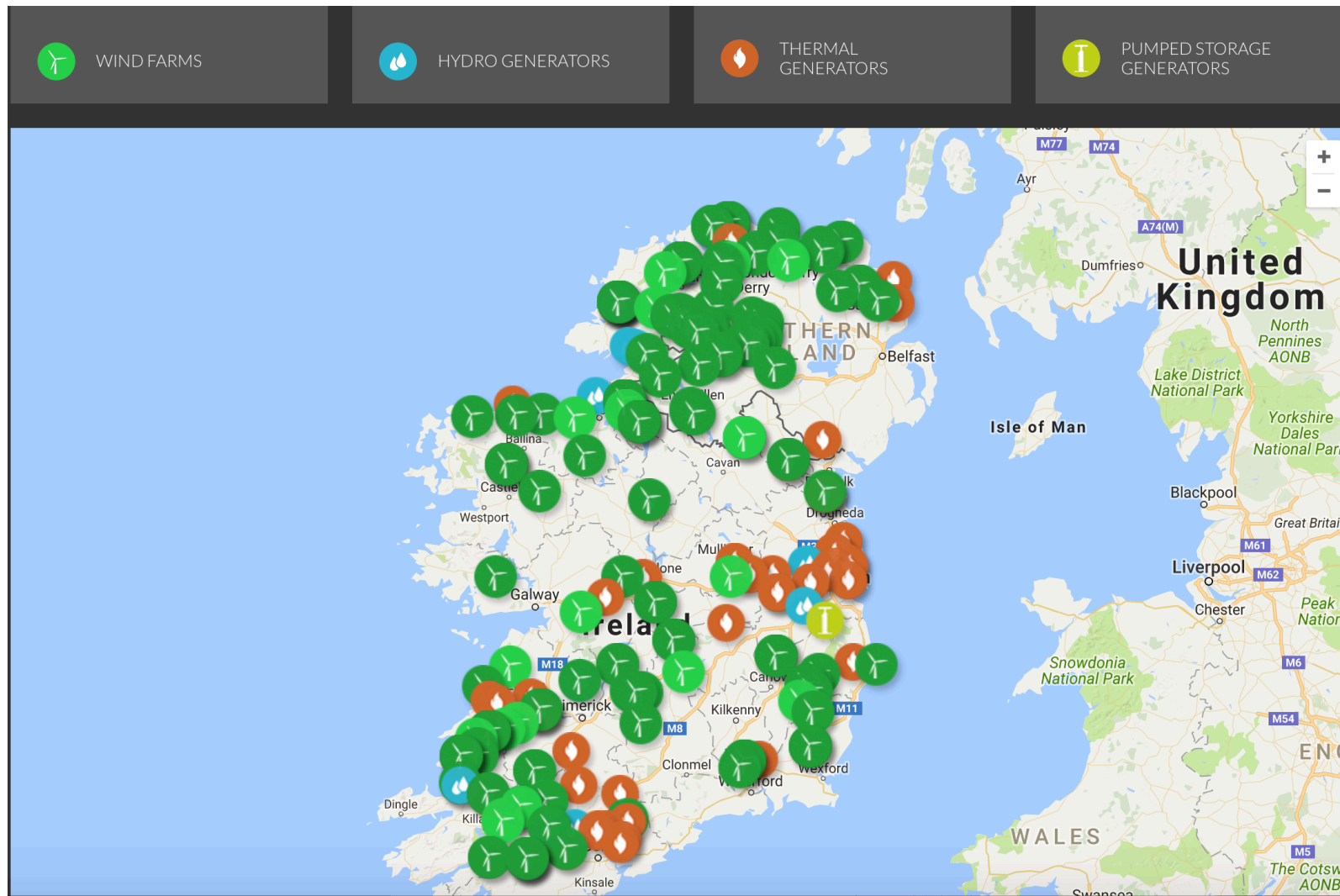
- 400kV Stations
- 275kV Stations
- 220kV Stations
- 110kV Stations

#### Transmission Connected Generation

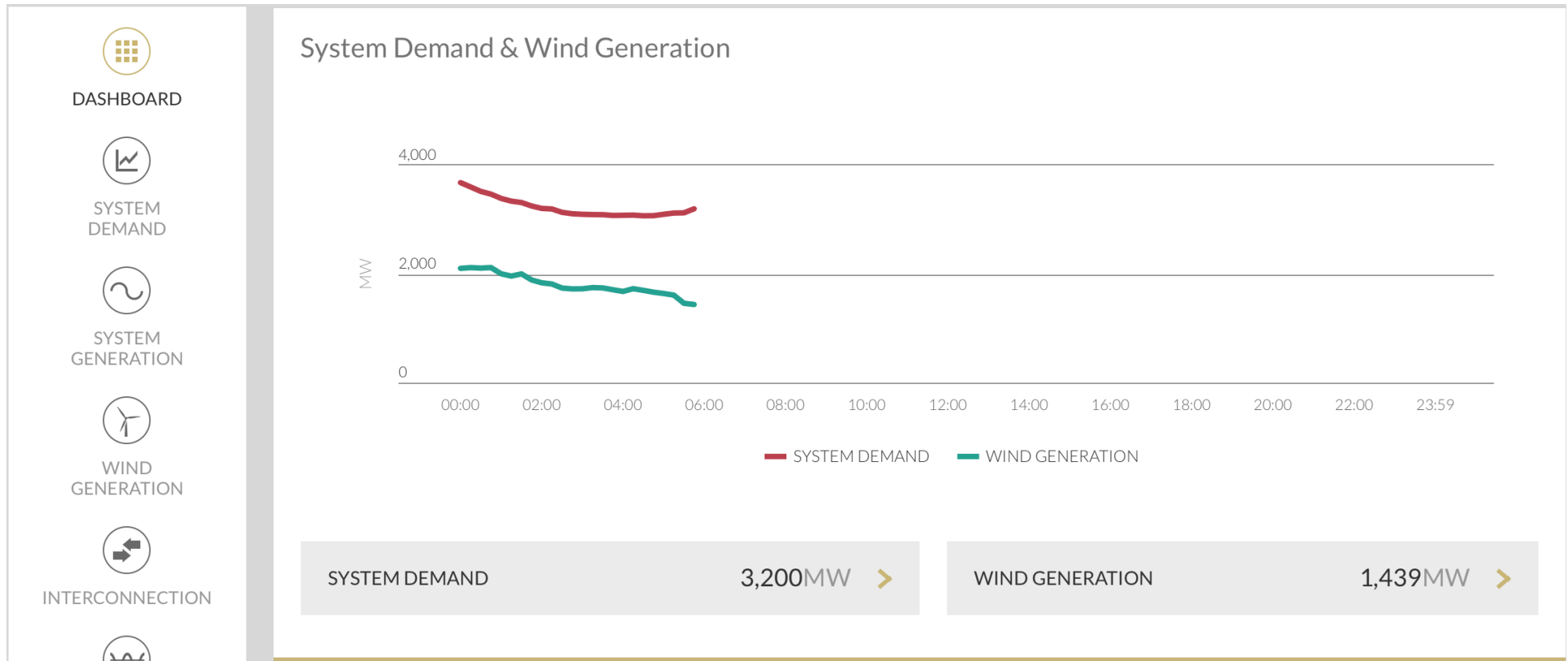
- Hydro Generation
- Thermal Generation
- ▼ Pumped Storage Generation
- Wind Generation



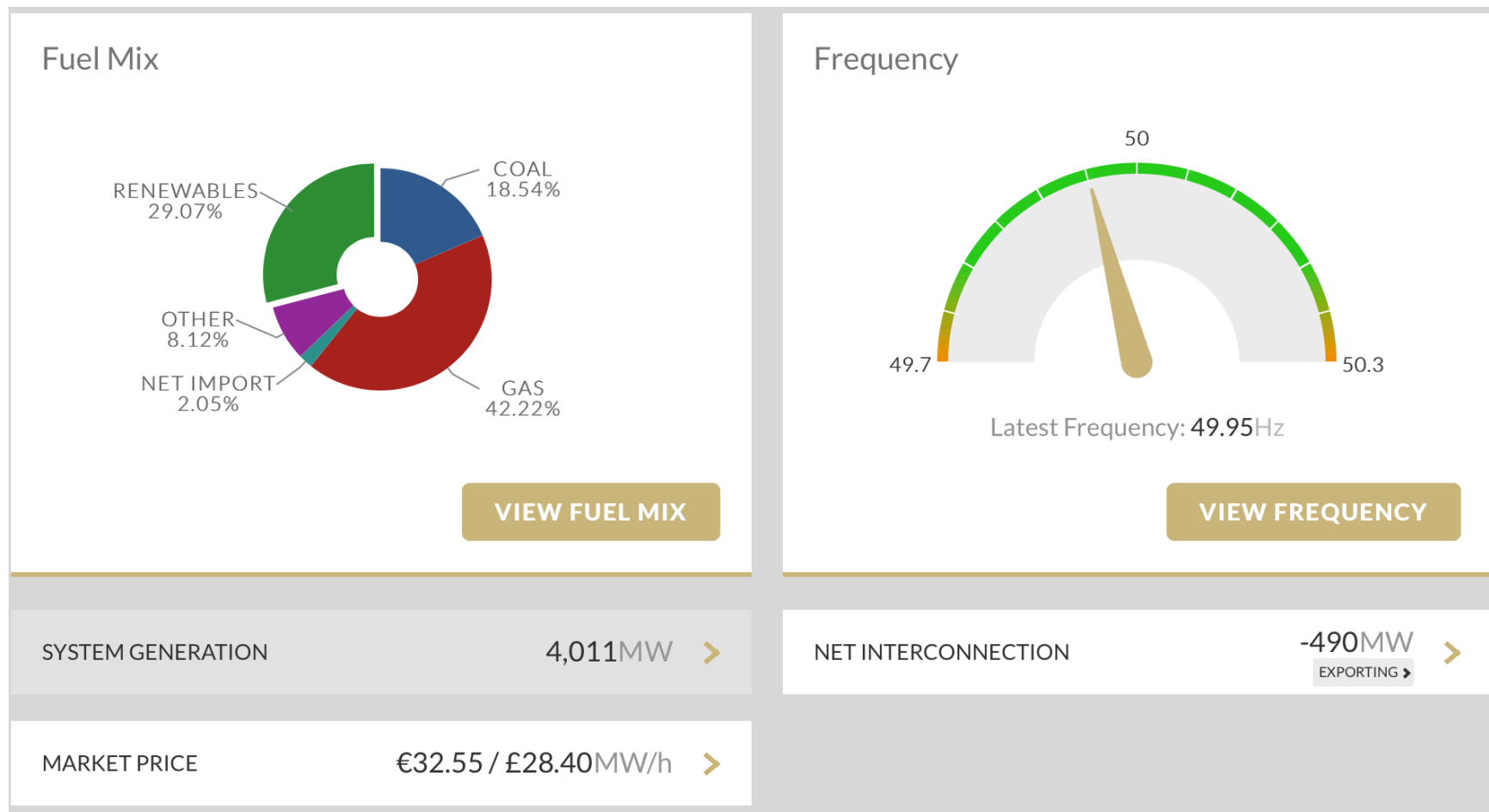
# Generation Information



<http://smartgriddashboard.eirgrid.com>



# Fuel Mix, Frequency & Net Flows



# Generation over time

## Actual System Generation

System Generation represents the total electricity production on the system, including system losses, but net of generators' requirements. System Generation is shown in 15 minute intervals.

DAY

WEEK

MONTH

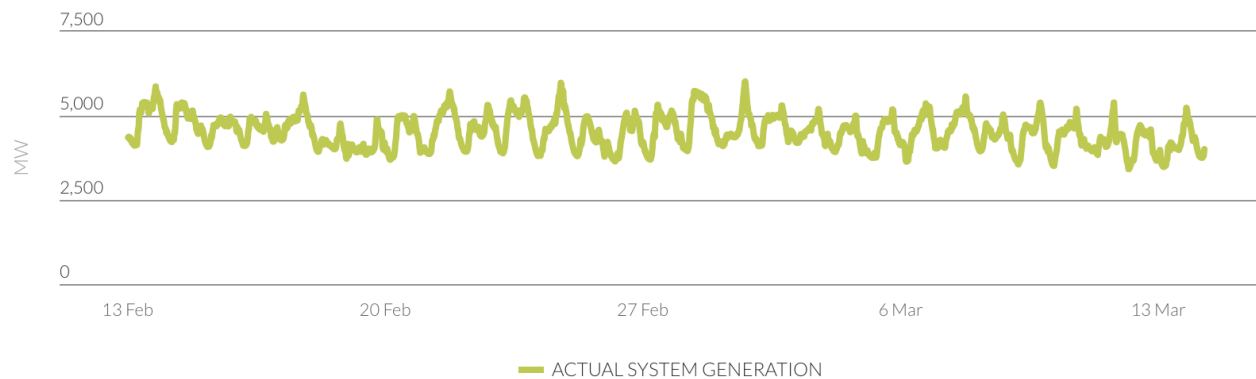
COMPARE WITH OTHER DATA



<

Last 30 Days (13/02/2017 - 14/03/2017)

>



# Extracting the data (download csv)

VIEW FULL BREAKDOWN IN TABLE

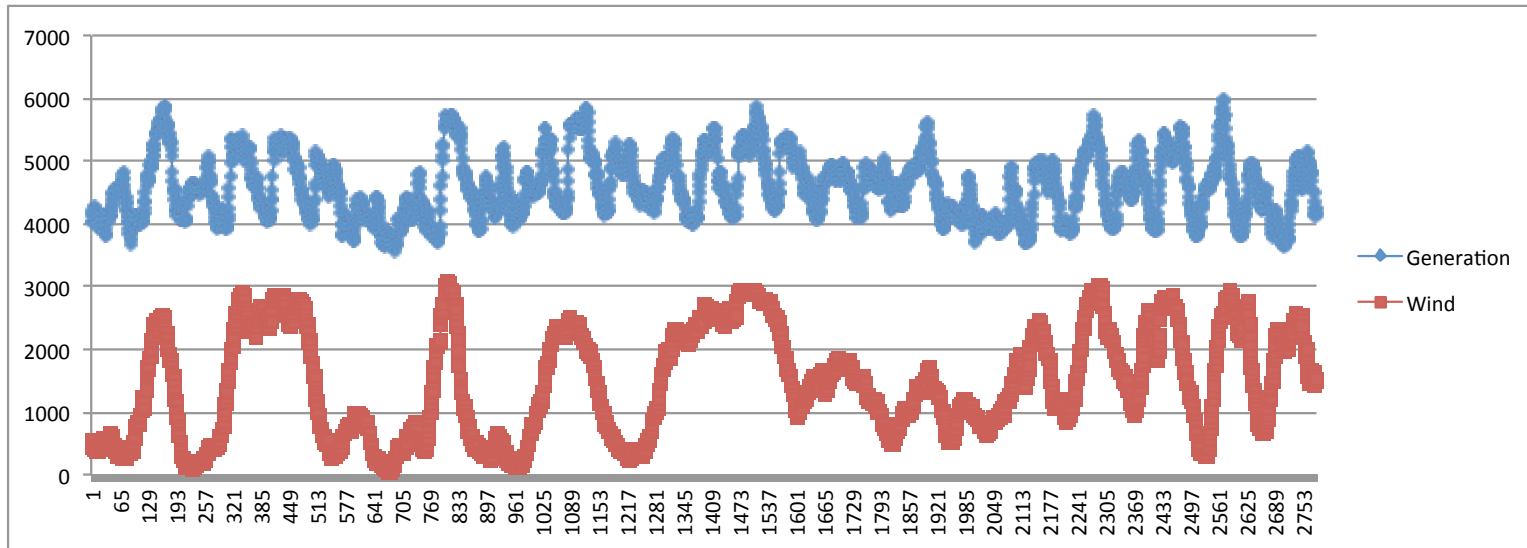


TIME	GENERATION (MW)
14 March 2017, 00:00	4,301
14 March 2017, 00:15	4,294
14 March 2017, 00:30	4,213
14 March 2017, 00:45	4,161
14 March 2017, 01:00	4,078
14 March 2017, 01:15	4,027
14 March 2017, 01:30	4,005



# Combine individual downloads

DateTime	Demand	Generation	Wind	CO2	NetImports	EWIC	Moyle
29/01/17 00:00	3834	4041	449	552	-145	-33	-112
29/01/17 00:15	3785	4041	505	548	-200	-108	-92
29/01/17 00:30	3708	4130	521	544	-294	-183	-111
29/01/17 00:45	3634	4181	492	543	-419	-258	-161
29/01/17 01:00	3581	4211	538	555	-503	-333	-170
29/01/17 01:15	3552	4278	561	531	-598	-379	-219
29/01/17 01:30	3491	4133	484	545	-516	-374	-142
29/01/17 01:45	3435	4143	474	551	-581	-365	-216
29/01/17 02:00	3374	4158	442	550	-653	-373	-280
29/01/17 02:15	3329	4135	421	550	-676	-377	-299
29/01/17 02:30	3299	4122	415	553	-693	-394	-299
29/01/17 02:45	3256	4146	401	558	-761	-469	-292





# Additional R Packages

Package	Purpose
gdata	Read excel files into R
lubridate	Process times and dates
tidyr	Split column information



# Get and Prepare Data for Analysis

```
library(gdata)
```

```
ener <- as_data_frame(read.xls("data/energy/data/IrelandData January 2017.xlsx"),  
                      stringsAsFactors=F)
```

```
> ener
```

```
# A tibble: 2,784 × 8
```

	DateTime	Demand	Generation	Wind	CO2	NetImports	EWIC	Moyle
	<fctr>	<int>	<int>	<int>	<int>	<int>	<int>	<int>
1	2017-01-29 00:00:00.00	3834	4041	449	552	-145	-33	-112
2	2017-01-29 00:15:00.00	3785	4041	505	548	-200	-108	-92
3	2017-01-29 00:30:00.00	3708	4130	521	544	-294	-183	-111
4	2017-01-29 00:45:00.00	3634	4181	492	543	-419	-258	-161
5	2017-01-29 01:00:00.00	3581	4211	538	555	-503	-333	-170
6	2017-01-29 01:15:00.00	3552	4278	561	531	-598	-379	-219
7	2017-01-29 01:30:00.00	3491	4133	484	545	-516	-374	-142
8	2017-01-29 01:45:00.00	3435	4143	474	551	-581	-365	-216
9	2017-01-29 02:00:00.00	3374	4158	442	550	-653	-373	-280
10	2017-01-29 02:15:00.00	3329	4135	421	550	-676	-377	-299

```
# ... with 2,774 more rows
```

# Parsing functions

Order of elements in date-time	Parse function
year, month, day	<code>ymd()</code>
year, day, month	<code>ydm()</code>
month, day, year	<code>mdy()</code>
day, month, year	<code>dmy()</code>
hour, minute	<code>hm()</code>
hour, minute, second	<code>hms()</code>
year, month, day, hour, minute, second	<code>ymd_hms()</code>

```
> dmy("12-01-2010")  
[1] "2010-01-12"  
>  
> ymd("2010-01-12")  
[1] "2010-01-12"
```

# Manipulating date-times

- Each element of a date-time object can be extracted
- Accessor functions allow this

Date component	Accessor
Year	<code>year()</code>
Month	<code>month()</code>
Week	<code>week()</code>
Day of year	<code>yday()</code>
Day of month	<code>mday()</code>
Day of week	<code>wday()</code>
Hour	<code>hour()</code>
Minute	<code>minute()</code>
Second	<code>second()</code>
Time zone	<code>tz()</code>

# dplyr Basics: 5 key functions

Function	Purpose
<b>filter()</b>	Pick observations by their values
<b>arrange()</b>	Reorder the rows
<b>select()</b>	Pick variables by their names
<b>mutate()</b>	Create new variables with functions of existing variables
<b>summarise()</b>	Collapse many values down to a single summary

# Extracting useful date info.

```
> ener
```

```
# A tibble: 2,784 × 8
```

```
DateTime [
```

```
<fctr>
```

```
1 2017-01-29 00:00:00.00
```

```
2 2017-01-29 00:15:00.00
```

```
3 2017-01-29 00:30:00.00
```

```
4 2017-01-29 00:45:00.00
```

```
5 2017-01-29 01:00:00.00
```

```
6 2017-01-29 01:15:00.00
```

```
7 2017-01-29 01:30:00.00
```

```
8 2017-01-29 01:45:00.00
```

```
9 2017-01-29 02:00:00.00
```

```
10 2017-01-29 02:15:00.00
```

```
# ... with 2,774 more rows
```

- Convert factor into date object
- Create new features:
  - Hour of day
  - Minute of day
  - Day of Week



# mutate() – Creates new features

```
ener <- mutate(ener, DateTime = ymd_hms(DateTime),  
                HourOfDay    = hour(DateTime),  
                MinuteOfDay   = minute(DateTime),  
                DayOfWeek     = wday(DateTime, label=T))
```

```
> ener
```

```
# A tibble: 2,784 × 11
```

	DateTime	Demand	Generation	Wind	CO2	NetImports	EWIC	Moyle	HourOfDay	MinuteOfDay	DayOfWeek
	<dtm>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<ord>
1	2017-01-29 00:00:00	3834	4041	449	552	-145	-33	-112	0	0	Sun
2	2017-01-29 00:15:00	3785	4041	505	548	-200	-108	-92	0	15	Sun
3	2017-01-29 00:30:00	3708	4130	521	544	-294	-183	-111	0	30	Sun
4	2017-01-29 00:45:00	3634	4181	492	543	-419	-258	-161	0	45	Sun
5	2017-01-29 01:00:00	3581	4211	538	555	-503	-333	-170	1	0	Sun
6	2017-01-29 01:15:00	3552	4278	561	531	-598	-379	-219	1	15	Sun
7	2017-01-29 01:30:00	3491	4133	484	545	-516	-374	-142	1	30	Sun
8	2017-01-29 01:45:00	3435	4143	474	551	-581	-365	-216	1	45	Sun
9	2017-01-29 02:00:00	3374	4158	442	550	-653	-373	-280	2	0	Sun
10	2017-01-29 02:15:00	3329	4135	421	550	-676	-377	-299	2	15	Sun

```
# ... with 2,774 more rows
```

# **tidyr** package – four fundamental functions of data tidying

- **gather()** takes multiple columns, and gathers them into key-value pairs: it makes “wide” data longer
- **spread()** takes two columns (key and value) and spreads into multiple columns, it makes long data wider
- **separate()** splits a single column into multiple columns
- **unite()** combines multiple columns into a single column



# separate()

[https://rpubs.com/bradleyboehmke/data\\_wrangling](https://rpubs.com/bradleyboehmke/data_wrangling)

```
Function:      separate(data, col, into, sep = " ", remove = TRUE, convert = FALSE)
Same as:       data %>% separate(col, into, sep = " ", remove = TRUE, convert = FALSE)
```

## Arguments:

```
data:          data frame
col:           column name representing current variable
into:          names of variables representing new variables
sep:           how to separate current variable (char, num, or symbol)
remove:        if TRUE, remove input column from output data frame
convert:       if TRUE will automatically convert values to logical, integer, numeric, complex or
               factor as appropriate
```

> unsep

	Date	Product	Sales
1	Jan_2005	ABC	1000
2	Jan_2006	DEF	2000
3	Jan_2007	ABC	3000

> separate(unsep, Date, c("Year", "Month"))

	Year	Month	Product	Sales
1	Jan	2005	ABC	1000
2	Jan	2006	DEF	2000
3	Jan	2007	ABC	3000



# Split date/time column

```
ener <- separate(ener, DateTime, c("Date", "Time"), sep=" ", remove=F)
```

```
> ener
```

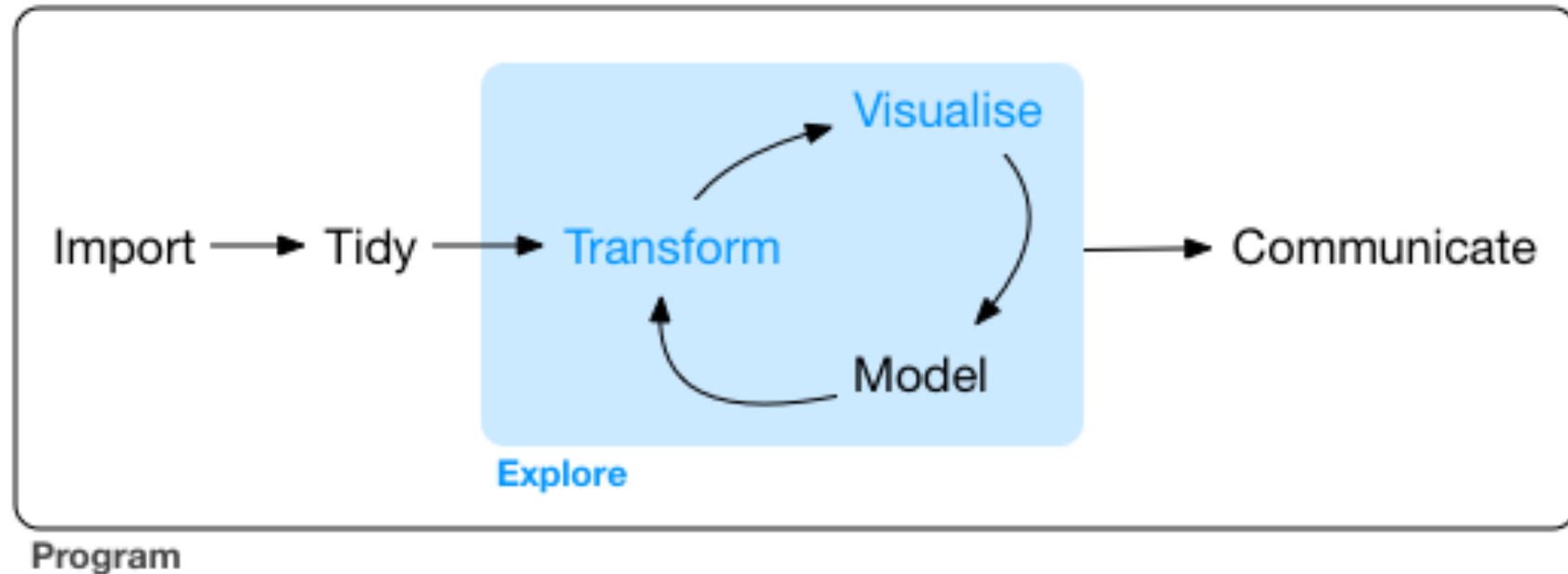
```
# A tibble: 2,784 × 13
```

	DateTime	Date	Time	Demand	Generation	Wind	CO2	NetImports	EWIC
*	<dtm>	<chr>	<chr>	<int>	<int>	<int>	<int>	<int>	<int>
1	2017-01-29 00:00:00	2017-01-29	00:00:00	3834	4041	449	552	-145	-33
2	2017-01-29 00:15:00	2017-01-29	00:15:00	3785	4041	505	548	-200	-108
3	2017-01-29 00:30:00	2017-01-29	00:30:00	3708	4130	521	544	-294	-183
4	2017-01-29 00:45:00	2017-01-29	00:45:00	3634	4181	492	543	-419	-258
5	2017-01-29 01:00:00	2017-01-29	01:00:00	3581	4211	538	555	-503	-333
6	2017-01-29 01:15:00	2017-01-29	01:15:00	3552	4278	561	531	-598	-379
7	2017-01-29 01:30:00	2017-01-29	01:30:00	3491	4133	484	545	-516	-374
8	2017-01-29 01:45:00	2017-01-29	01:45:00	3435	4143	474	551	-581	-365
9	2017-01-29 02:00:00	2017-01-29	02:00:00	3374	4158	442	550	-653	-373
10	2017-01-29 02:15:00	2017-01-29	02:15:00	3329	4135	421	550	-676	-377

```
# ... with 2,774 more rows, and 4 more variables: Moyle <int>, HourOfDay <int>,  
# MinuteOfDay <int>, DayOfWeek <ord>
```

# Data Exploration

“Data exploration is the art of looking at your data, rapidly generating hypotheses, quickly testing them, then repeating again and again and again.” (Wickham and Grolemund 2017).

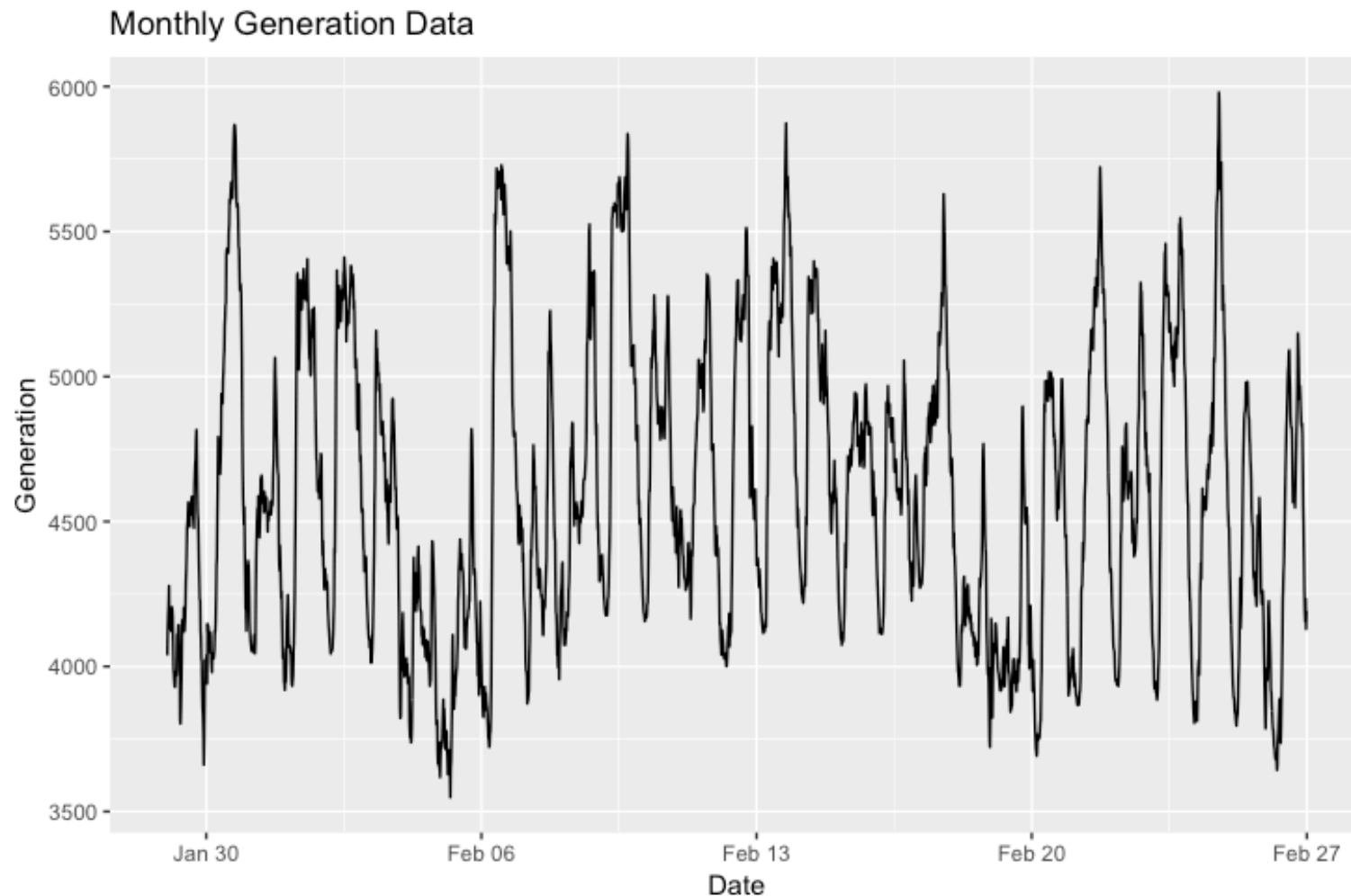


# Exploring Data Relationships

- Time series:
  - Generation data over the entire month
  - Generation data over a day
- Time of Day v Net Imports
- Demand v Net Imports
- Wind Generation v CO2 Emissions
- Comparing Wind Generation with Overall Generation

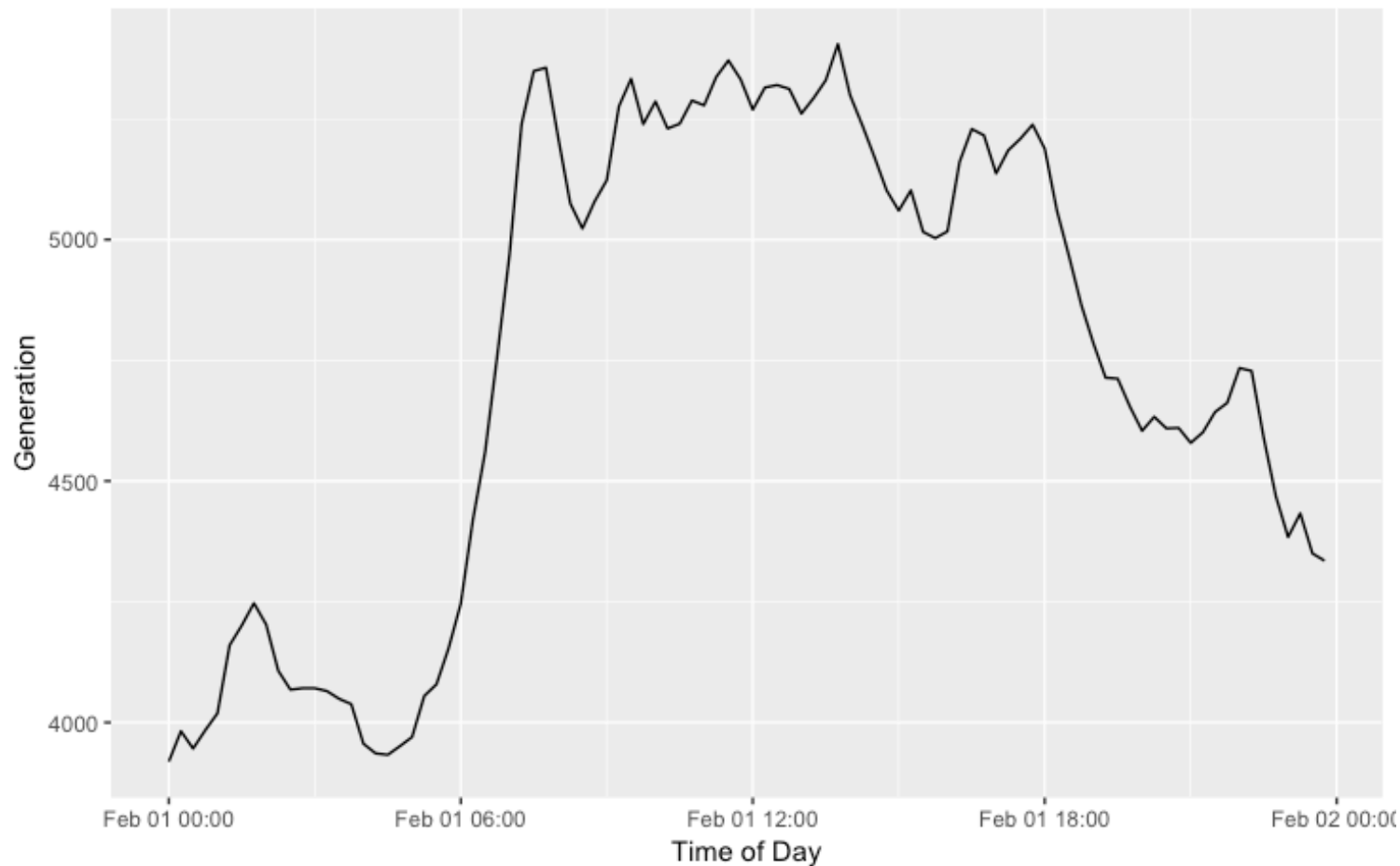


```
ggplot(data = ener) +  
  geom_line(mapping = aes(x=DateTime,y=Generation)) +  
  xlab("Date") + ylab("Generation") + ggtitle("Monthly Generation Data")
```

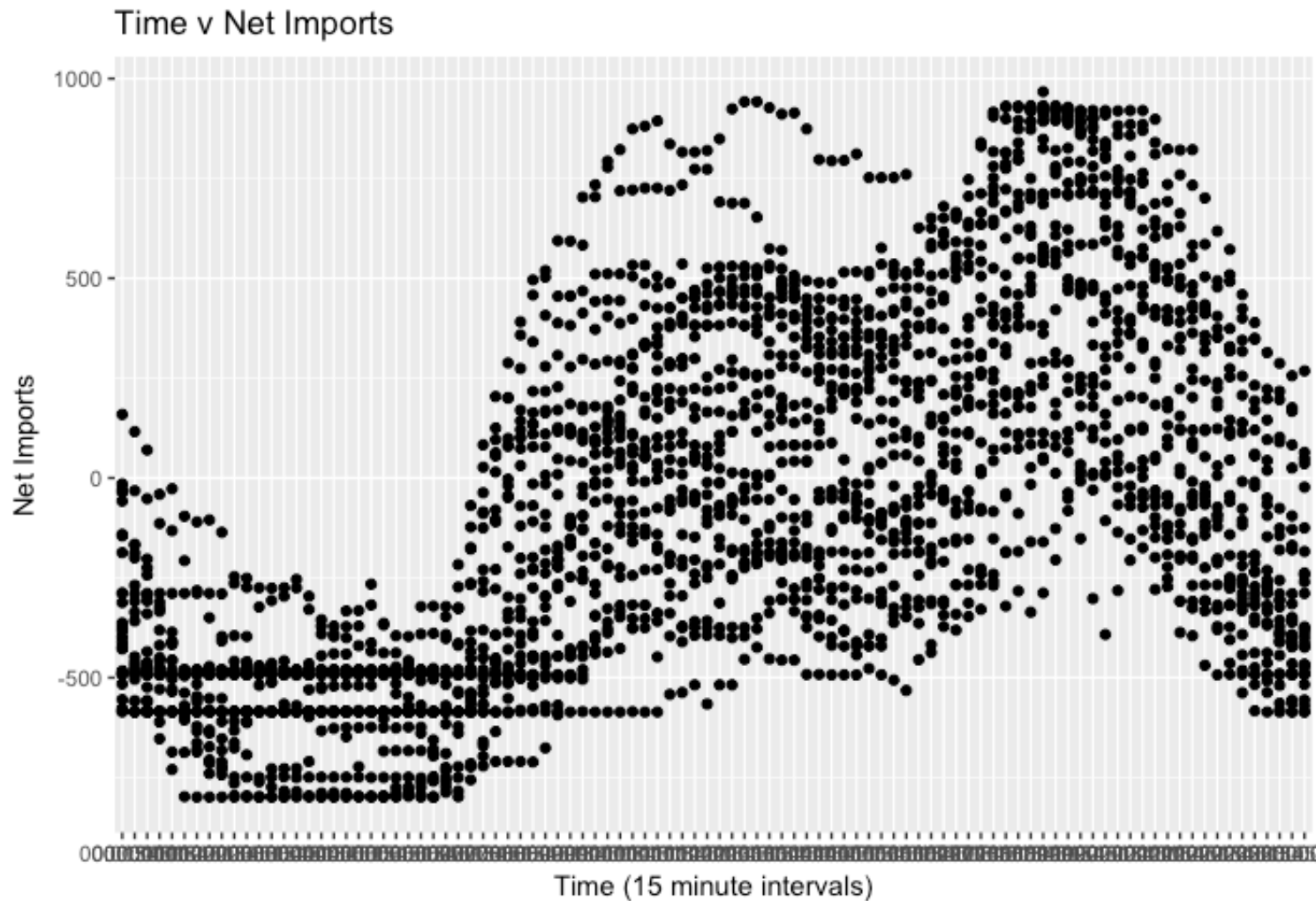


```
ggplot(data = filter(ener, Date=="2017-02-01")) +  
  geom_line(mapping = aes(x=DateTime, y=Generation)) +  
  xlab("Time of Day") + ylab("Generation") +  
  ggtitle("Generation Data for 1st Feb 2017")
```

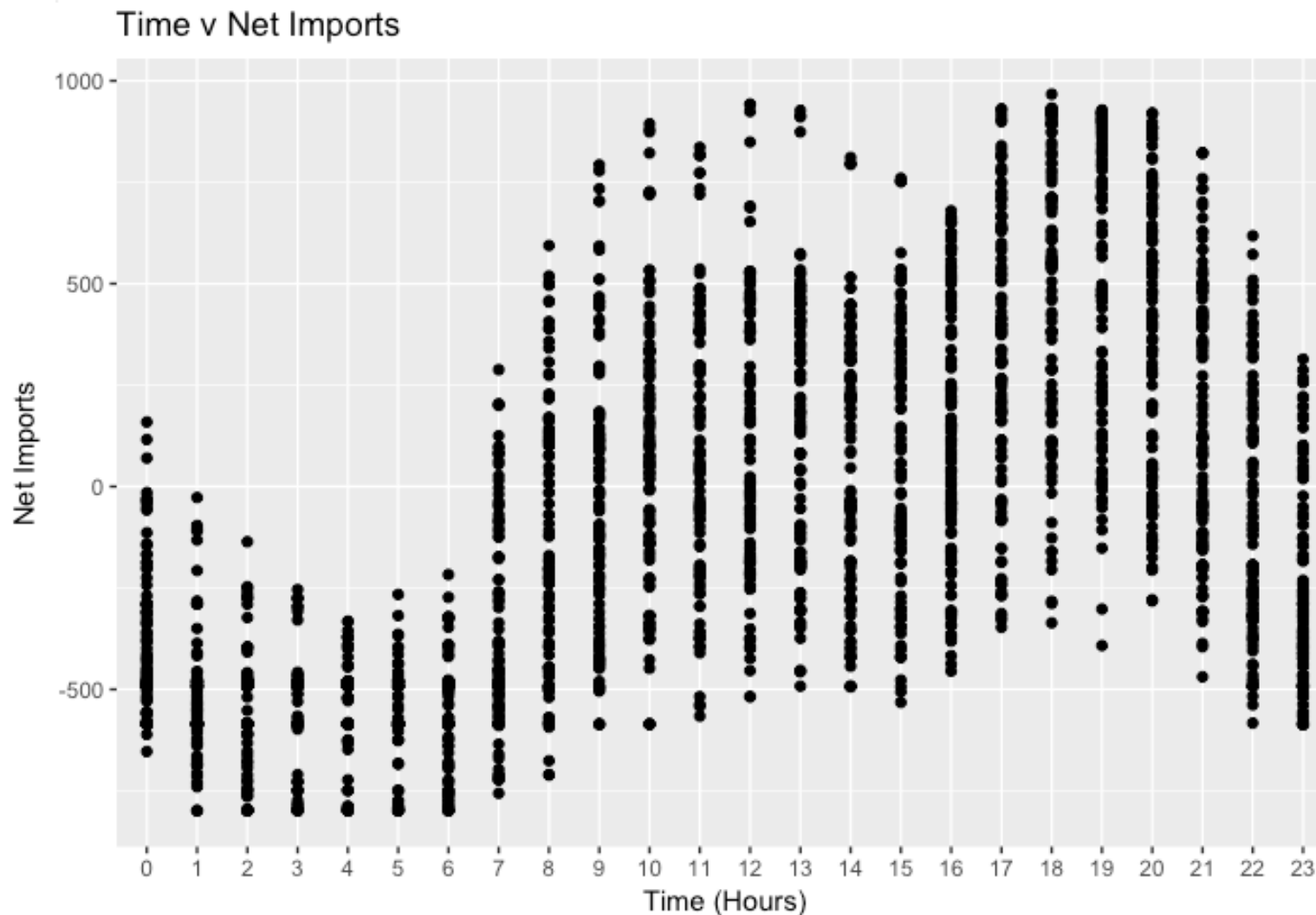
Generation Data for 1st Feb 2017



```
ggplot(data = ener) +  
  geom_point(mapping = aes(x=as.factor(Time),y=NetImports)) +  
  xlab("Time (15 minute intervals)") + ylab("Net Imports") +  
  ggtitle("Time v Net Imports")
```

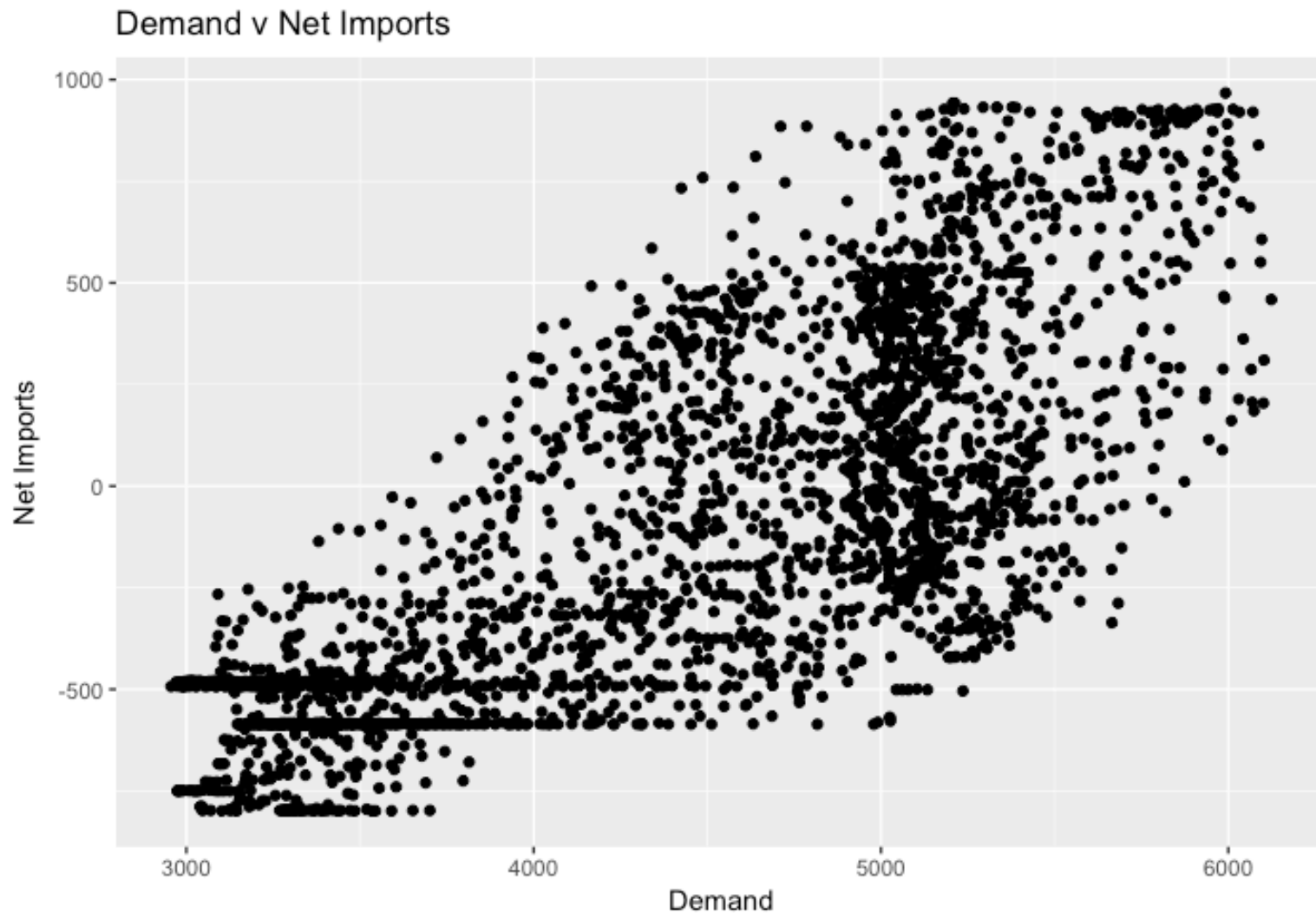


```
ggplot(data = ener) +
  geom_point(mapping = aes(x=as.factor(HourOfDay),y=NetImports)) +
  xlab("Time (Hours)") + ylab("Net Imports") +
  ggtitle("Time v Net Imports")
```

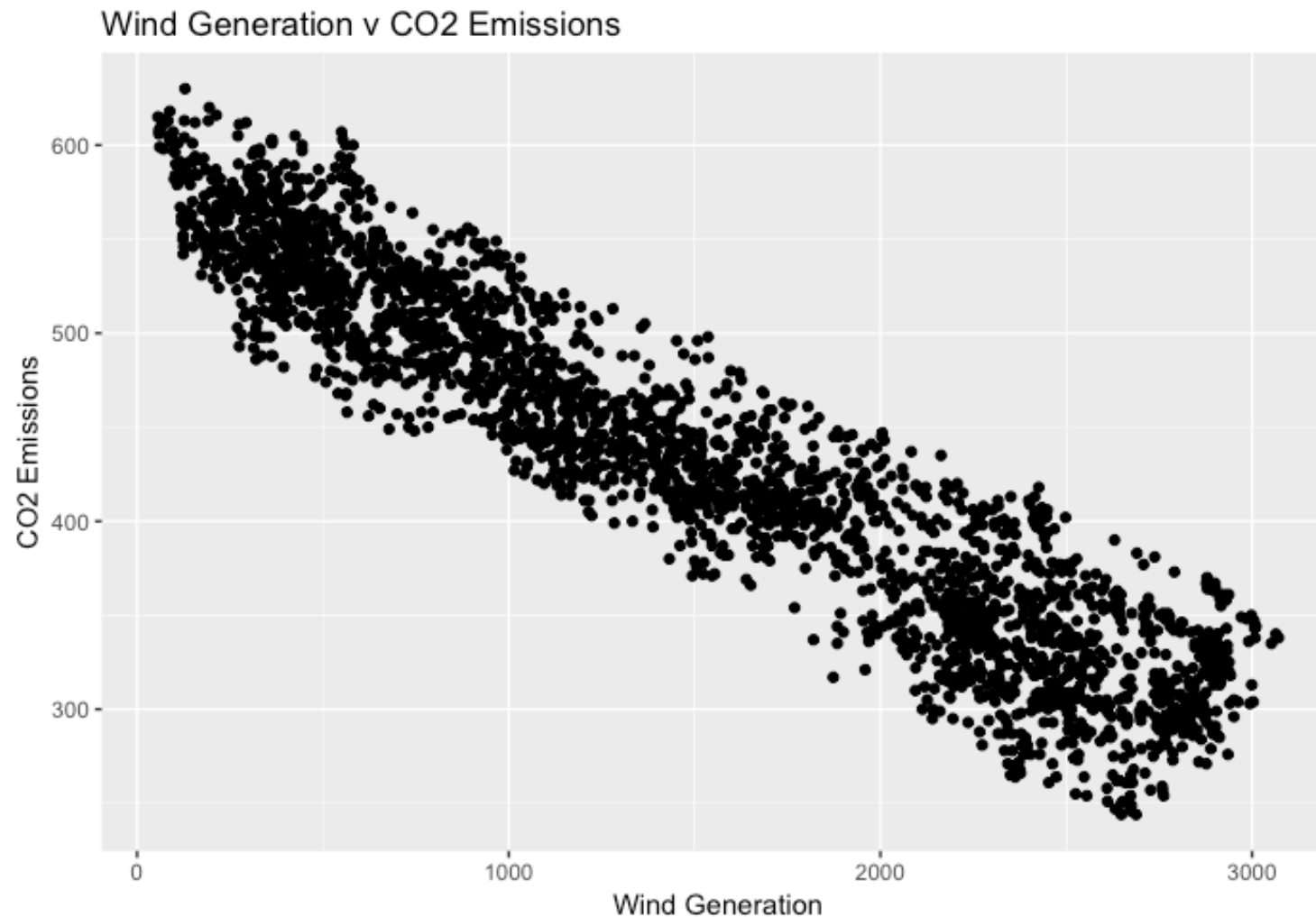




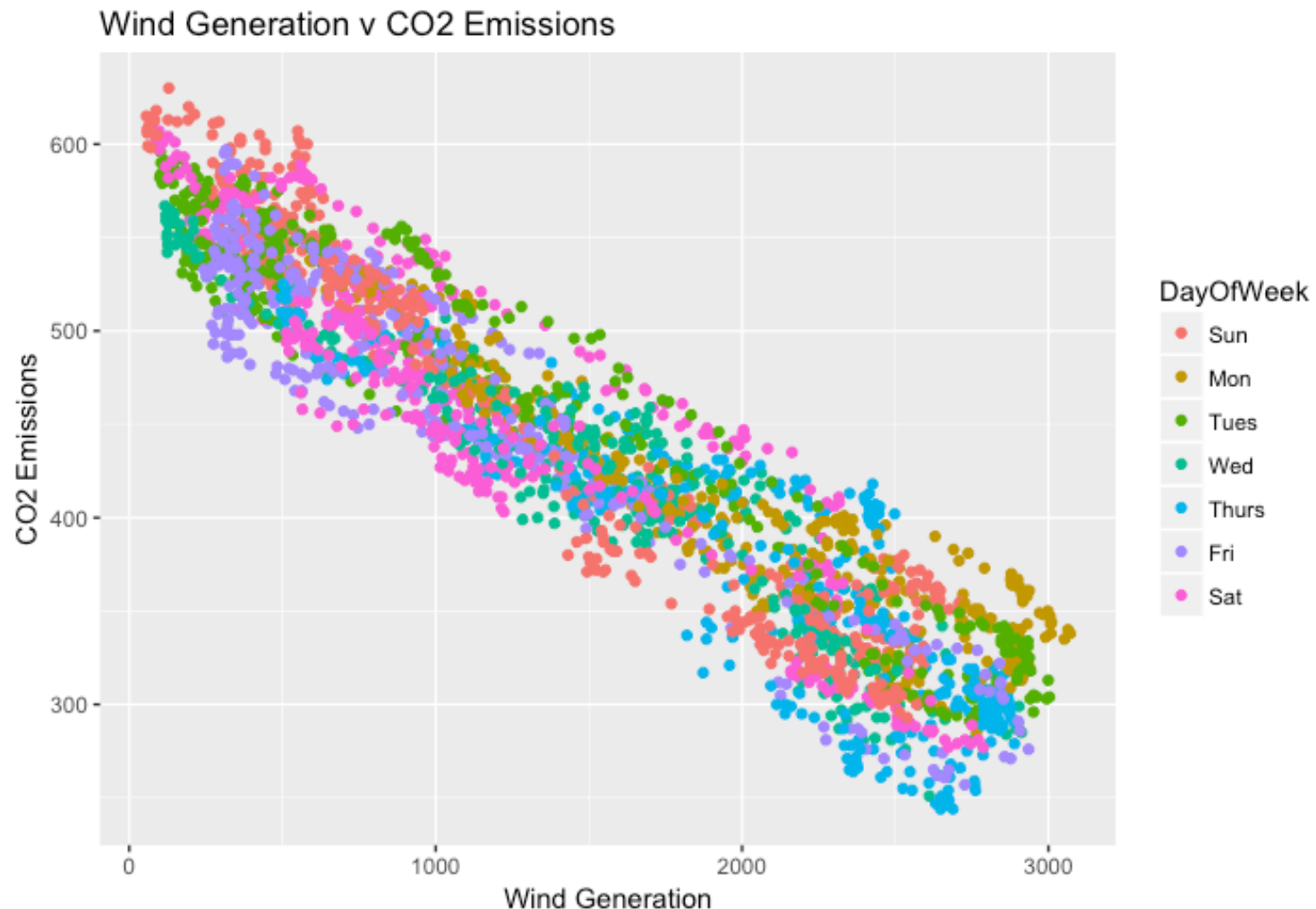
```
ggplot(data = ener) +  
  geom_point(mapping = aes(x=Demand,y=NetImports)) +  
  xlab("Demand") + ylab("Net Imports") +  
  ggtitle("Demand v Net Imports")
```



```
ggplot(data = ener) +  
  geom_point(mapping = aes(x=Wind,y=CO2)) +  
  xlab("Wind Generation") + ylab("CO2 Emissions") +  
  ggtitle("Wind Generation v CO2 Emissions")
```



```
ggplot(data = ener) +  
  geom_point(mapping = aes(x=Wind,y=C02,colour=DayOfWeek)) +  
  xlab("Wind Generation") + ylab("C02 Emissions") +  
  ggtitle("Wind Generation v C02 Emissions")
```



# Summary

- Identify data source
- Import data and prepare for analysis
- Using ggplot to explore hypotheses
- Next steps
  - dplyr
  - Integrate weather data

