wt\_data.R

sumedh

Mon Oct 08 16:29:41 2018

# Setting the Working Directory  
setwd("C:/Users/sumedh/Desktop/assignment")  
# Loading the DataSet

# NOTE DATASET is rename to ‘wt\_data.csv’ from wtgdata.csv  
wtdata=read.csv("wt\_data.csv")  
library(WindCurves)# For drawing the Power curves

## Warning: package 'WindCurves' was built under R version 3.5.1

#libraries  
library(MASS)  
library(tidyverse)

## Warning: package 'tidyverse' was built under R version 3.5.1

## -- Attaching packages ---------------------------------- tidyverse 1.2.1 --

## v ggplot2 2.2.1 v purrr 0.2.5  
## v tibble 1.4.2 v dplyr 0.7.5  
## v tidyr 0.8.1 v stringr 1.3.1  
## v readr 1.1.1 v forcats 0.3.0

## -- Conflicts ------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()  
## x dplyr::select() masks MASS::select()

library(knitr)

## Warning: package 'knitr' was built under R version 3.5.1

library(viridis)

## Loading required package: viridisLite

library(dplyr)  
library(scales)

## Warning: package 'scales' was built under R version 3.5.1

##   
## Attaching package: 'scales'

## The following object is masked from 'package:viridis':  
##   
## viridis\_pal

## The following object is masked from 'package:purrr':  
##   
## discard

## The following object is masked from 'package:readr':  
##   
## col\_factor

View(wtdata)  
#Checking the Structure of the data set  
str(wtdata)

## 'data.frame': 61269 obs. of 11 variables:  
## $ unitlocation : Factor w/ 2 levels "WTG01","WTG02": 1 1 1 1 1 1 1 1 1 1 ...  
## $ ttimestamplocal: Factor w/ 30702 levels "01-01-14 0:00",..: 2 3 4 5 6 7 8 9 10 11 ...  
## $ power : num 964 1025 1180 1754 1827 ...  
## $ windspeed : num 9.1 9.6 9.6 11.2 11.4 10.9 10.3 9.1 10.5 11 ...  
## $ airdensity : num 1.16 1.16 1.16 1.16 1.16 ...  
## $ amb\_temp : int 27 27 27 27 27 27 27 27 27 27 ...  
## $ nac\_direction : num 61.8 61.8 61.8 65.2 66.6 70.7 70 59.8 67.1 68.4 ...  
## $ blds\_pitchangle: num -3 -3.1 -2.9 -1.5 -1.3 -1.5 -2.7 -2.6 -2.3 -2 ...  
## $ rtr\_rpm : num 14.1 14.4 14.4 14.5 14.5 14.5 14.5 13.8 14.5 14.5 ...  
## $ gen\_rpm : num 1305 1336 1330 1345 1344 ...  
## $ wtg\_state : Factor w/ 8 levels "curtailed","data error",..: 6 6 6 6 6 6 6 6 6 6 ...

# By seeing the structure of the dataset ttimestamplocal is in factor , It should be change into the Date Time Format  
summary(wtdata)

## unitlocation ttimestamplocal power windspeed   
## WTG01:30591 01-01-14 0:00: 2 Min. : -28.3 Min. : 0.000   
## WTG02:30678 01-06-13 0:00: 2 1st Qu.: 208.7 1st Qu.: 5.300   
## 01-06-13 0:10: 2 Median : 623.6 Median : 7.600   
## 01-06-13 0:20: 2 Mean : 766.5 Mean : 7.491   
## 01-06-13 0:30: 2 3rd Qu.:1284.3 3rd Qu.: 9.900   
## 01-06-13 0:40: 2 Max. :2000.4 Max. :20.800   
## (Other) :61257 NA's :3 NA's :3   
## airdensity amb\_temp nac\_direction blds\_pitchangle   
## Min. :1.132 Min. :20.00 Min. : 0.00 Min. :-4.400   
## 1st Qu.:1.154 1st Qu.:28.00 1st Qu.: 63.00 1st Qu.:-2.700   
## Median :1.158 Median :28.00 Median : 80.30 Median :-2.400   
## Mean :1.157 Mean :28.38 Mean : 90.04 Mean : 3.775   
## 3rd Qu.:1.158 3rd Qu.:29.00 3rd Qu.:101.90 3rd Qu.: 0.400   
## Max. :1.190 Max. :35.00 Max. :359.00 Max. :90.200   
## NA's :3 NA's :3   
## rtr\_rpm gen\_rpm wtg\_state   
## Min. : 0.00 Min. : 0 ok :44020   
## 1st Qu.: 8.90 1st Qu.: 825 wind low : 8658   
## Median :12.40 Median :1151 curtailed : 4187   
## Mean :10.82 Mean :1004 Gen not conn : 3897   
## 3rd Qu.:14.50 3rd Qu.:1340 service : 475   
## Max. :14.60 Max. :1347 high wind cut-out: 28   
## NA's :3 NA's :3 (Other) : 4

## Exploratory Data analysis (EDA) using the DataExplorer package  
library(DataExplorer)# Data Explorer package is called for data analysis

## Warning: package 'DataExplorer' was built under R version 3.5.1

library(data.table) # supporting library for the Data Explorer

##   
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':  
##   
## between, first, last

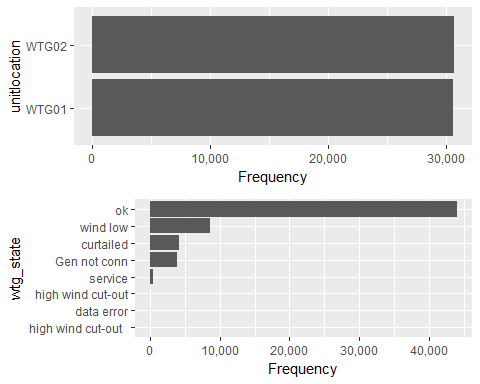
## The following object is masked from 'package:purrr':  
##   
## transpose

introduce(wtdata)

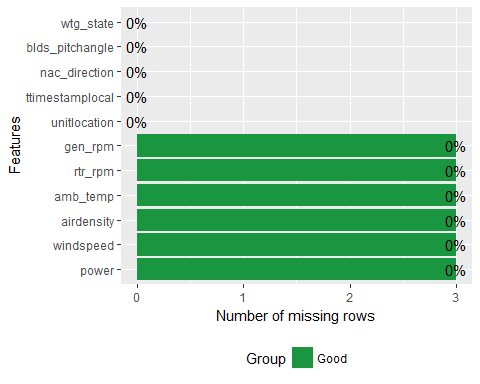
## rows columns discrete\_columns continuous\_columns all\_missing\_columns  
## 1 61269 11 3 8 0  
## total\_missing\_values total\_observations memory\_usage  
## 1 18 673959 6625872

plot\_str(wtdata) # Plot the attribute datatype.  
plot\_bar(wtdata)

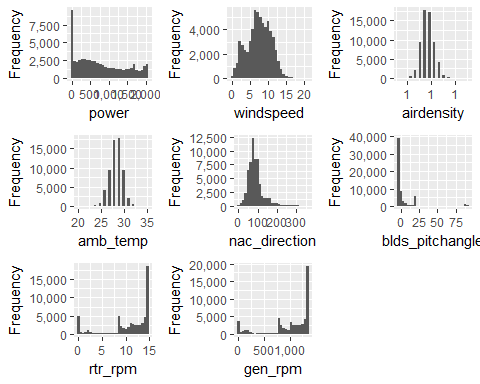
## 1 columns ignored with more than 50 categories.  
## ttimestamplocal: 30702 categories



# From the graph it is clear that number of observation for Wind turbine (WT) is almost equal  
# Wind turbine of "OK" stage is having the highest frequency.  
plot\_missing(wtdata)

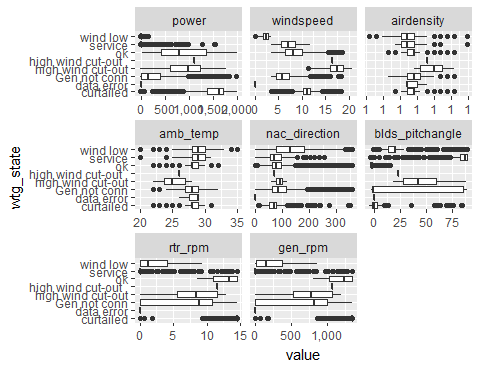


# There are 3 missing values having column name is gen\_rpm,rtr\_rpm,amb\_temp,windspeed, power  
plot\_histogram(wtdata)



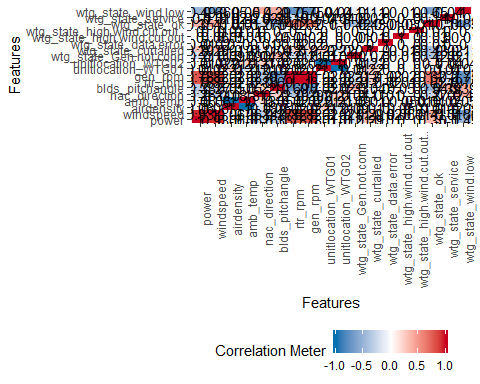
# Power is spread is from 0 to 2000.  
# Wind Speed is distributed like weibull Distribution curve.  
# Air Density also distributed like Normal Distribution curve.  
# amb\_temp is distributed like Normal Distribution.  
# nac\_direction is right skewed which means that most of the values is in between 50 to 150.  
# rotor and generator rpm follow the left skew distribution.  
# Most of the blade pitch angle between -1 to +25 degree.  
  
plot\_boxplot(wtdata,by="wtg\_state")

## Warning: Removed 18 rows containing non-finite values (stat\_boxplot).

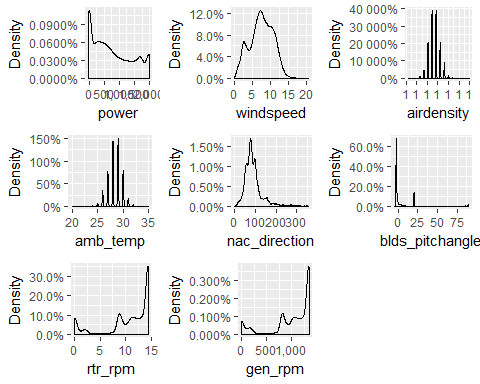


wtdata=na.omit(wtdata)# Ommiting the missing value in the dataset because in the wtg\_state "service" and "get not conn" condition data was not generated.  
plot\_correlation(wtdata)

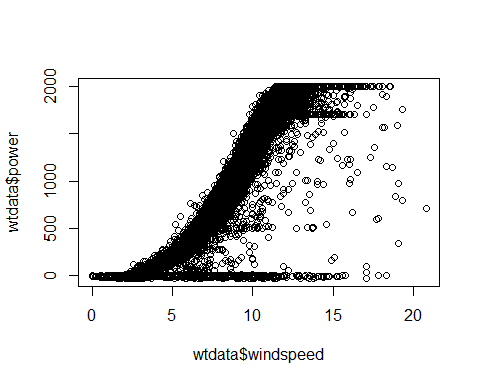
## 1 features with more than 20 categories ignored!  
## ttimestamplocal: 30702 categories



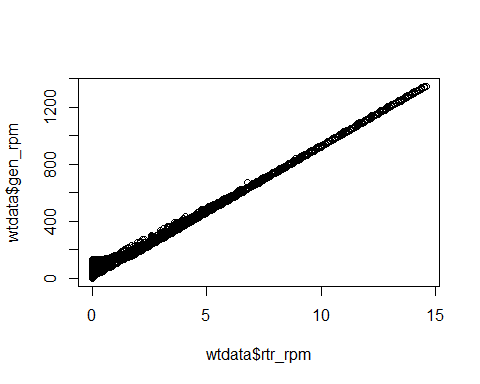
# power with windspeed is highly correlated after that generator and rotar speed.  
# With Windspeed also rotor and generator rpm is correlated  
# Wtg\_state\_ok is correlated with the rotor and generator rpm.  
plot\_density(wtdata)



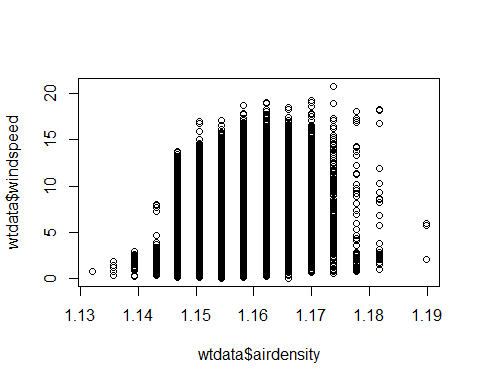
# The density curve also the same pattern as th histogram curve is showing.  
plot(wtdata$windspeed,wtdata$power) # Scatter plot between windspeed vs power which is looks like S shape curve.



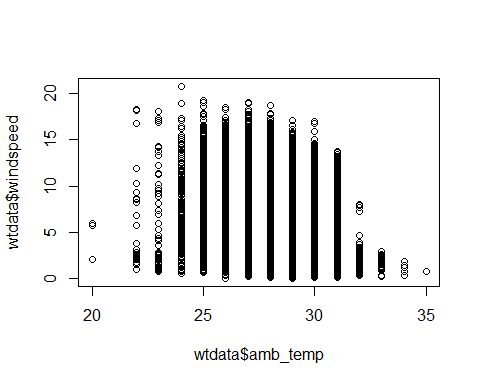
plot(wtdata$rtr\_rpm,wtdata$gen\_rpm)# rotor rmp and generator rpm is correlated.



plot(wtdata$airdensity,wtdata$windspeed)# As Air density increases wind speed descreases



plot(wtdata$amb\_temp,wtdata$windspeed)# Atmosphere temp. in between 25 to 30 show good result for wind speed.



library(data.table)  
wtdata\_dt=data.table(wtdata)  
  
a=group\_category(wtdata\_dt,"wtg\_state",threshold = 0.05)  
print(a)

## wtg\_state cnt pct cum\_pct  
## 1: ok 44020 0.71850619 0.7185062  
## 2: wind low 8658 0.14131819 0.8598244  
## 3: curtailed 4187 0.06834133 0.9281657

#From this it is clear that "ok" percentage of data is around 71.85% in the whole dataset  
output=split\_columns(wtdata)  
# There are three discrete columns and 8 continous columns and zero missing column  
output$num\_discrete

## [1] 3

output$num\_continuous

## [1] 8

output$num\_all\_missing

## [1] 0

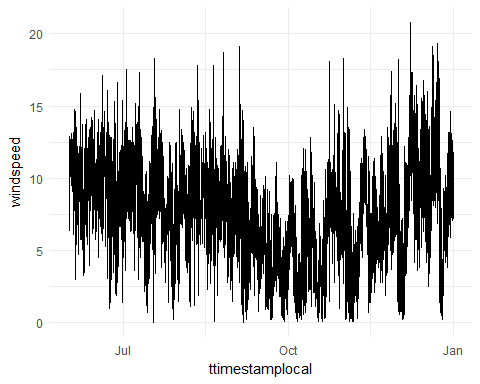
## Converting the timestamp data in character format ot data time format with the help lubridate package.  
library(lubridate)

##   
## Attaching package: 'lubridate'

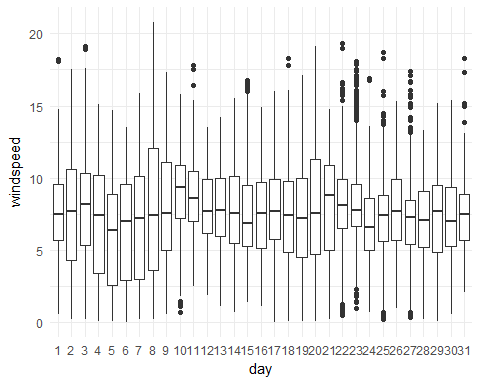
## The following objects are masked from 'package:data.table':  
##   
## hour, isoweek, mday, minute, month, quarter, second, wday,  
## week, yday, year

## The following object is masked from 'package:base':  
##   
## date

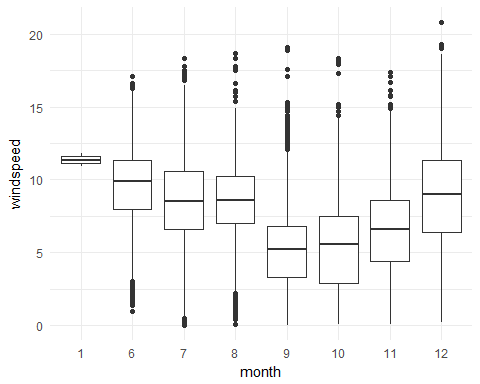
wtdata<-mutate(wtdata,ws\_norm=windspeed/max(windspeed),power\_norm=power/max(power))  
wtdata$ttimestamplocal=dmy\_hm(wtdata$ttimestamplocal)  
library(ggplot2)  
ggplot(wtdata,aes(ttimestamplocal,windspeed))+geom\_line()+theme\_minimal()



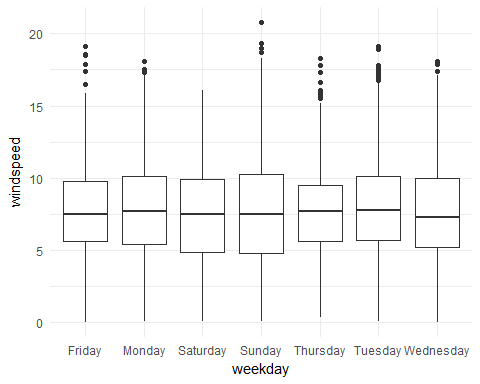
#### From the graph it is clear that Avg. wind speed is deceasing from June to October and then started increasing from October to January.  
wtdata$month=month(wtdata$ttimestamplocal)  
wtdata$year=year(wtdata$ttimestamplocal)  
wtdata$day=day(wtdata$ttimestamplocal)  
wtdata$date=date(wtdata$ttimestamplocal)  
wtdata$weekday=weekdays.POSIXt(wtdata$ttimestamplocal)  
  
ggplot(wtdata,aes(factor(day),windspeed))+  
 geom\_boxplot()+  
 theme\_minimal()+  
 labs(x='day')



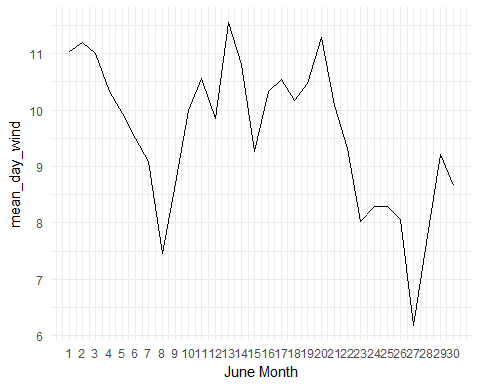
#ggplot(wtdata,aes(day,windspeed))+geom\_line()+theme\_minimal()  
###############################################  
ggplot(wtdata,aes(factor(month),windspeed))+  
 geom\_boxplot()+  
 theme\_minimal()+  
 labs(x='month')



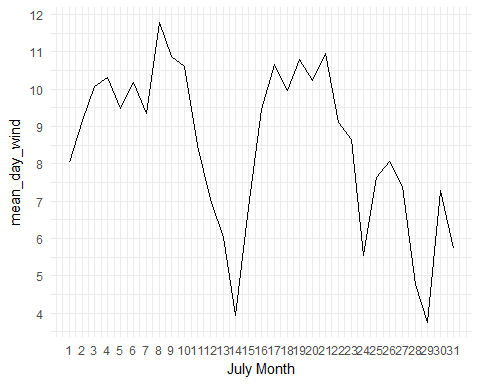
#ggplot(wtdata,aes(month,windspeed))+geom\_line()+theme\_minimal()  
################################################  
  
ggplot(wtdata,aes(factor(weekday),windspeed))+  
 geom\_boxplot()+  
 theme\_minimal()+  
 labs(x='weekday')



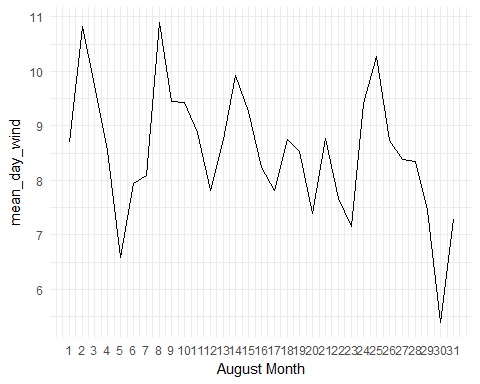
#ggplot(wtdata,aes(weekday,windspeed))+geom\_line()+theme\_minimal()  
  
#Ques 3 Ques 3 Use the fitted curve to create the difference measured data and the fit.  
# A) Is there any change point over time?  
# B) Can you Estimate the Point of Change?  
## For June data.  
june=wtdata %>% filter(month==6) %>% select(month,day,windspeed)%>% group\_by(day)%>% summarize(mean\_day\_wind=mean(windspeed))  
   
ggplot(june,aes(day,mean\_day\_wind))+  
 geom\_line()+theme\_minimal()+labs(x='day')+labs(x='June Month')+scale\_x\_continuous(breaks=seq(1,30,1))+scale\_y\_continuous(breaks=seq(1,12,1))



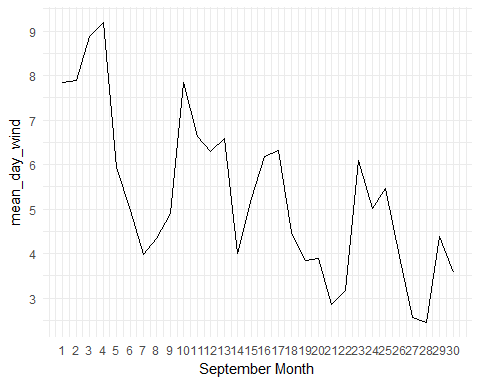
# June month graph is clear that there is two sudden dip on 8th June and 27th June.  
# On 8th June value is around average windspeed 7.5m/s on 27th 6.2 m/s.  
  
## For July data.  
july=wtdata %>% filter(month==7) %>% select(month,day,windspeed)%>% group\_by(day)%>% summarize(mean\_day\_wind=mean(windspeed))  
  
ggplot(july,aes(day,mean\_day\_wind))+  
 geom\_line()+theme\_minimal()+labs(x='day')+labs(x='July Month')+scale\_x\_continuous(breaks=seq(1,31,1))+scale\_y\_continuous(breaks=seq(1,12,1))



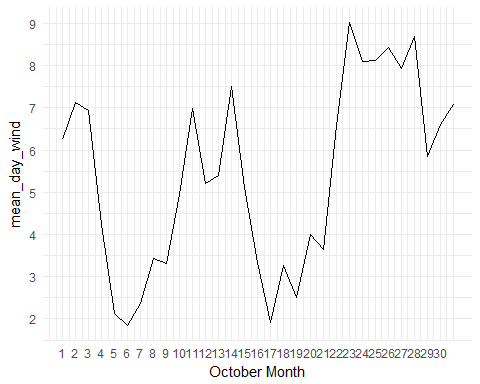
## In July Month THere is a sudden dip on 14th July and 29th July and mean windspeed values are in between 3 m/s to 4 m/s.  
  
## For August data.  
aug=wtdata %>% filter(month==8) %>% select(month,day,windspeed)%>% group\_by(day)%>% summarize(mean\_day\_wind=mean(windspeed))  
ggplot(aug,aes(day,mean\_day\_wind))+  
 geom\_line()+theme\_minimal()+labs(x='day')+labs(x='August Month')+scale\_x\_continuous(breaks=seq(1,31,1))+scale\_y\_continuous(breaks=seq(1,12,1))



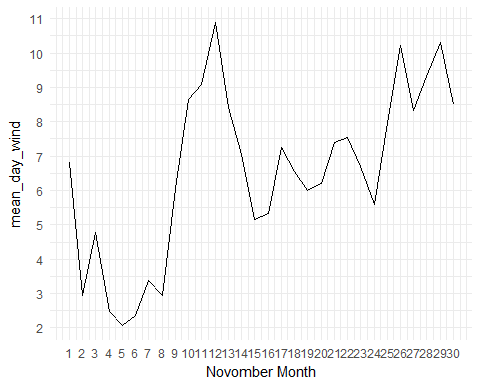
## There is the sudden dip in the mean windspeed on 5th and 30th day of August Month. On 5th value is around 6.5m/s and on 30th around 5.4 m/s   
  
## For September data.  
sep=wtdata %>% filter(month==9) %>% select(month,day,windspeed)%>% group\_by(day)%>% summarize(mean\_day\_wind=mean(windspeed))  
ggplot(sep,aes(day,mean\_day\_wind))+  
 geom\_line()+theme\_minimal()+labs(x='day')+labs(x='September Month')+scale\_x\_continuous(breaks=seq(1,30,1))+scale\_y\_continuous(breaks=seq(1,12,1))



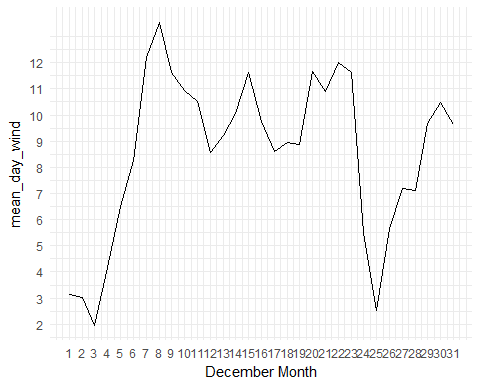
# There is a descresing trends for whole September month and on 28th of September lowest mean day wind.   
  
## For October data.  
oct=wtdata %>% filter(month==10) %>% select(month,day,windspeed)%>% group\_by(day)%>% summarize(mean\_day\_wind=mean(windspeed))  
ggplot(oct,aes(day,mean\_day\_wind))+  
 geom\_line()+theme\_minimal()+labs(x='day')+labs(x='October Month')+scale\_x\_continuous(breaks=seq(1,30,1))+scale\_y\_continuous(breaks=seq(1,12,1))



# From 1st Oct to 06th Oct mean wind speed is descreasing fro 6th oct to 12th Oct increasing from there again descreasing 17th oct then increases till 24 t Oct.  
  
## For November data.  
nov=wtdata %>% filter(month==11) %>% select(month,day,windspeed)%>% group\_by(day)%>% summarize(mean\_day\_wind=mean(windspeed))  
ggplot(nov,aes(day,mean\_day\_wind))+  
 geom\_line()+theme\_minimal()+labs(x='day')+labs(x='Novomber Month')+scale\_x\_continuous(breaks=seq(1,30,1))+scale\_y\_continuous(breaks=seq(1,12,1))



# In November on 6th and 17 th mean wind speed is lowest in whole month of mean\_day\_wind  
  
## For December Data  
dec=wtdata %>% filter(month==12) %>% select(month,day,windspeed)%>% group\_by(day)%>% summarize(mean\_day\_wind=mean(windspeed))  
ggplot(dec,aes(day,mean\_day\_wind))+  
 geom\_line()+theme\_minimal()+labs(x='day')+labs(x='December Month')+scale\_x\_continuous(breaks=seq(1,31,1))+scale\_y\_continuous(breaks=seq(1,12,1))



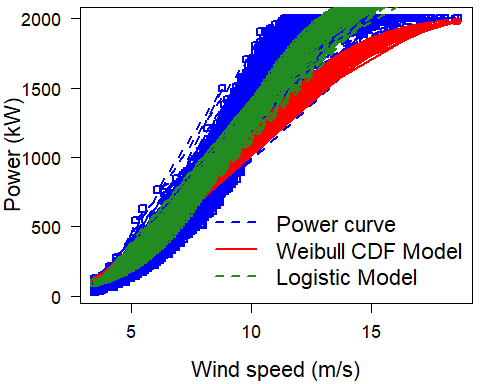
# On 3rd and 25th of December month there is a sudden dip in the data and the values in between 2 and 3 m/s of avg. day of the wind.  
  
  
#== Ques 1.Evaluate the Power curve for all States and state "ok"==#  
  
########### wtg\_state=='ok'##############  
ok=wtdata[wtdata$wtg\_state=='ok',]  
s\_ok=ok$windspeed  
p\_ok=ok$power  
da\_ok=data.frame(s\_ok,p\_ok)  
x\_ok=fitcurve(da\_ok)

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = 2.72185   
## Scale (C) = 10.69945   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 2266.124   
## phi 2 = 9.292145   
## phi 3 = 1.859689   
## ===================================

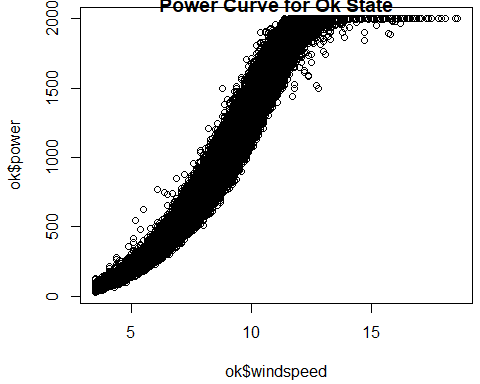
validate.curve(x\_ok)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 192.8286083 94.3113142  
## 2 MAE 138.6771501 73.3549201  
## 3 MAPE 16.4799120 9.9374798  
## 4 R2 0.8895457 0.9735779  
## 5 COR 0.9831741 0.9867061

plot(x\_ok)



plot(ok$windspeed,ok$power, main="Power Curve for Ok State")



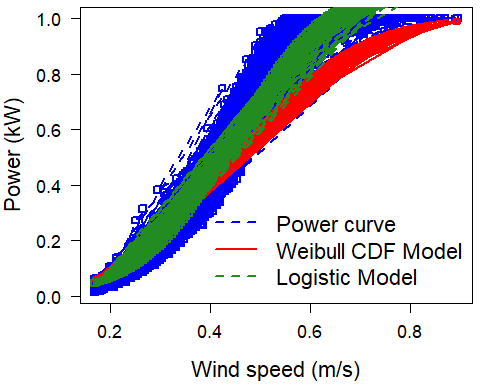
s\_okk=ok$ws\_norm  
p\_okk=ok$power\_norm  
da\_okk=data.frame(s\_okk,p\_okk)  
x\_okk=fitcurve(da\_okk)

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = 2.72185   
## Scale (C) = 0.5143968   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 1.132835   
## phi 2 = 0.4467377   
## phi 3 = 0.08940814   
## ===================================

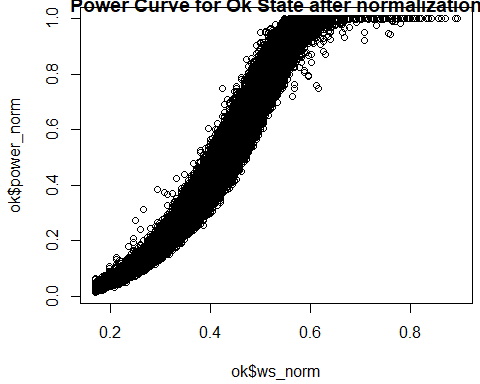
validate.curve(x\_okk)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 0.09639503 0.04714623  
## 2 MAE 0.06932471 0.03667013  
## 3 MAPE 16.47991198 9.93747984  
## 4 R2 0.88954567 0.97357790  
## 5 COR 0.98317406 0.98670607

plot(x\_okk)



plot(ok$ws\_norm,ok$power\_norm, main="Power Curve for Ok State after normalization")



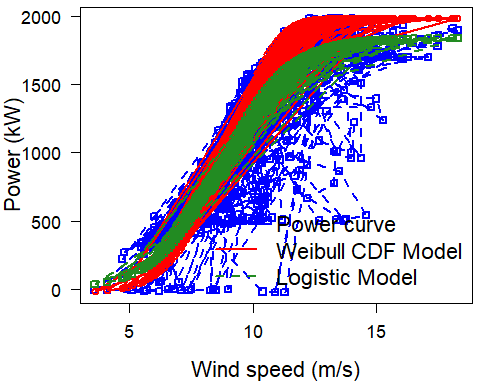
######### wtg\_state=='Curtailed' #########  
curtailed=wtdata[wtdata$wtg\_state=='curtailed',]  
s\_curtailed=curtailed$windspeed  
p\_curtailed=curtailed$power  
da\_curtailed=data.frame(s\_curtailed,p\_curtailed)  
x\_curtailed=fitcurve(da\_curtailed)

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = 6.030393   
## Scale (C) = 9.434009   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 1839.477   
## phi 2 = 8.740585   
## phi 3 = 1.321247   
## ===================================

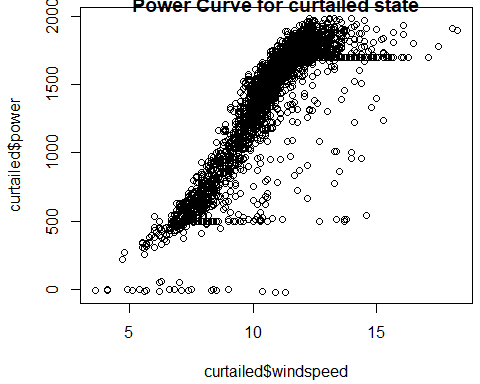
validate.curve(x\_curtailed)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 269.2124809 152.8335915  
## 2 MAE 227.6625127 89.8700986  
## 3 MAPE 19.1922232 8.1923393  
## 4 R2 0.5295469 0.8483775  
## 5 COR 0.9240383 0.9213246

plot(x\_curtailed)



plot(curtailed$windspeed,curtailed$power, main="Power Curve for curtailed state")



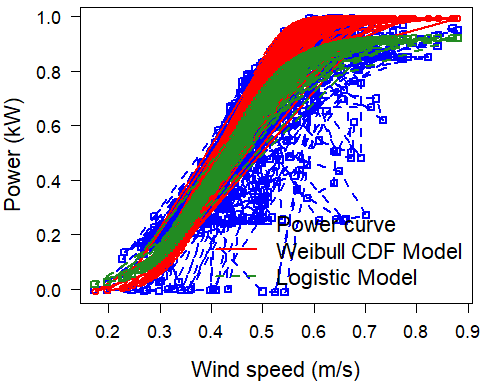
s\_curtailedd=curtailed$ws\_norm  
p\_curtailedd=curtailed$power\_norm  
da\_curtailedd=data.frame(s\_curtailedd,p\_curtailedd)  
x\_curtailedd=fitcurve(da\_curtailedd)

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = 6.030393   
## Scale (C) = 0.4535581   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 0.9195548   
## phi 2 = 0.4202204   
## phi 3 = 0.06352148   
## ===================================

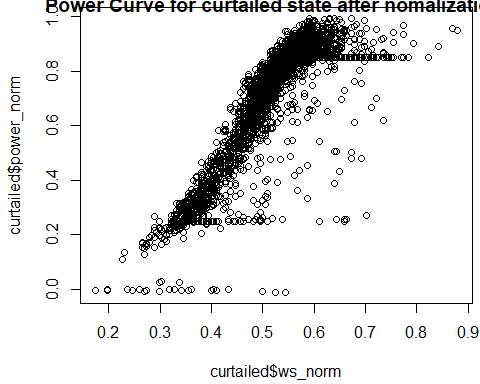
validate.curve(x\_curtailedd)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 0.1345793 0.07640152  
## 2 MAE 0.1138085 0.04492606  
## 3 MAPE 19.1922232 8.19233933  
## 4 R2 0.5295469 0.84837753  
## 5 COR 0.9240383 0.92132463

plot(x\_curtailedd)



plot(curtailed$ws\_norm,curtailed$power\_norm, main="Power Curve for curtailed state after nomalization ")



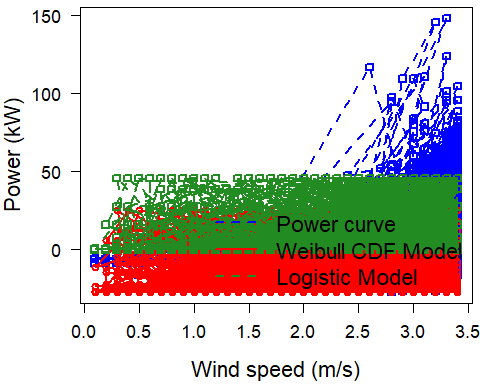
####### wtd\_state=='wind low' ##########  
windlow=wtdata[wtdata$wtg\_state=='wind low',]  
s\_windlow=windlow$windspeed  
p\_windlow=windlow$power  
da\_windlow=data.frame(s\_windlow,p\_windlow)  
x\_windlow=fitcurve(da\_windlow)

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = 5.787359   
## Scale (C) = 4.062834   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 51.57372   
## phi 2 = 3.185769   
## phi 3 = 0.1059129   
## ===================================

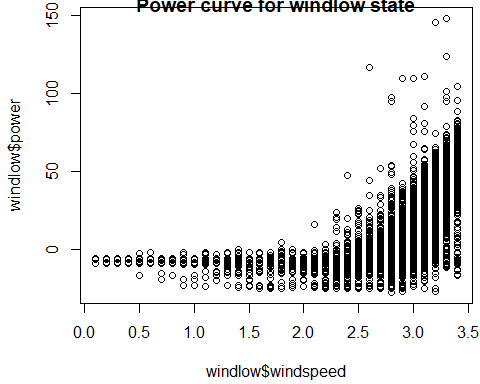
validate.curve(x\_windlow)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 24.940330 2.009099e+01  
## 2 MAE 19.535457 1.406907e+01  
## 3 MAPE 65.083473 4.299920e+11  
## 4 R2 -0.746007 -1.330378e-01  
## 5 COR 0.295026 2.792139e-01

plot(x\_windlow)



plot(windlow$windspeed,windlow$power,main="Power curve for windlow state")



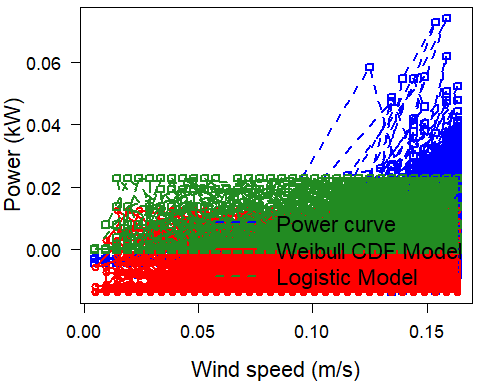
s\_windloww=windlow$ws\_norm  
p\_windloww=windlow$power\_norm  
da\_windloww=data.frame(s\_windloww,p\_windloww)  
x\_windloww=fitcurve(da\_windloww)

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = 5.787359   
## Scale (C) = 0.1953285   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 0.0257817   
## phi 2 = 0.153162   
## phi 3 = 0.005091967   
## ===================================

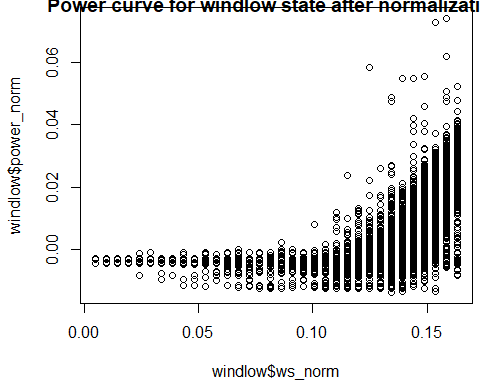
validate.curve(x\_windloww)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 0.012467671 1.004349e-02  
## 2 MAE 0.009765775 7.033129e-03  
## 3 MAPE 65.083472762 4.299920e+11  
## 4 R2 -0.746007046 -1.330378e-01  
## 5 COR 0.295026040 2.792139e-01

plot(x\_windloww)



plot(windlow$ws\_norm,windlow$power\_norm,main="Power curve for windlow state after normalization")



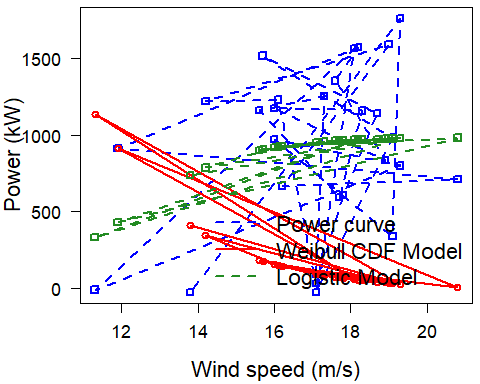
####### wtd\_stage=='high wind cut-out' ######  
windhigh=wtdata[wtdata$wtg\_state=='high wind cut-out',]  
s\_windhigh=windhigh$windspeed  
p\_windhigh=windhigh$power  
da\_windhigh=data.frame(s\_windhigh,p\_windhigh)  
x\_windhigh=fitcurve(da\_windhigh)

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = -6.600544   
## Scale (C) = 11.38474   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 982.0785   
## phi 2 = 12.2519   
## phi 3 = 1.403037   
## ===================================

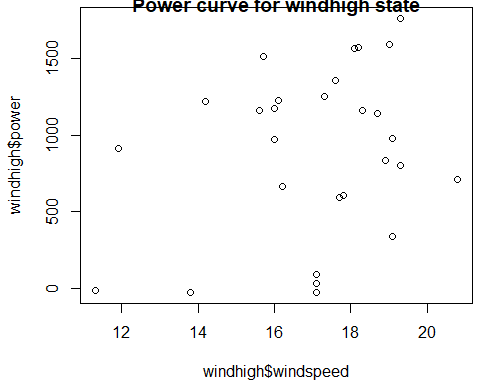
validate.curve(x\_windhigh)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 979.6625552 504.61060715  
## 2 MAE 855.1191057 432.43930613  
## 3 MAPE 1697.3991378 51.10013881  
## 4 R2 -2.4286377 0.09033533  
## 5 COR -0.3052457 0.30068561

plot(x\_windhigh)



plot(windhigh$windspeed,windhigh$power,main="Power curve for windhigh state")



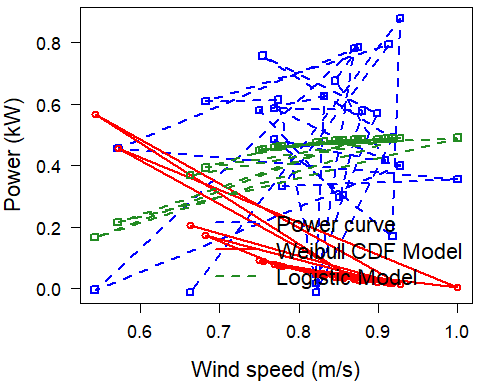
s\_windhighh=windhigh$ws\_norm  
p\_windhighh=windhigh$power\_norm  
da\_windhighh=data.frame(s\_windhighh,p\_windhighh)  
x\_windhighh=fitcurve(da\_windhighh)

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = -6.600544   
## Scale (C) = 0.5473432   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 0.4909411   
## phi 2 = 0.5890335   
## phi 3 = 0.06745372   
## ===================================

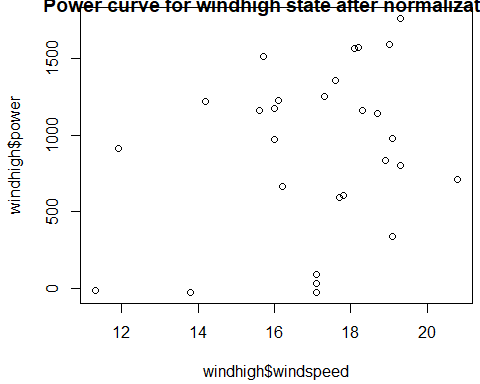
validate.curve(x\_windhighh)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 0.4897333 0.25225485  
## 2 MAE 0.4274741 0.21617642  
## 3 MAPE 1697.3991378 51.10013881  
## 4 R2 -2.4286377 0.09033533  
## 5 COR -0.3052457 0.30068561

plot(x\_windhighh)



plot(windhigh$windspeed,windhigh$power,main="Power curve for windhigh state after normalization")



####### wtd\_stage=='data error' ######  
#dataerror=wtdata[wtdata$wtg\_state=='data error',]  
#s\_dataerror=dataerror$windspeed  
#p\_dataerror=dataerror$power  
#da\_dataerror=data.frame(s\_dataerror,p\_dataerror)  
#x\_dataerror=fitcurve(da\_dataerror)  
#validate.curve(x\_dataerror)  
#plot(x\_dataerror)   
#" Could not able to plot the fitcurve because observation is too low i.e. 3"  
  
**#### Ques 2 Fit the curve to the two data and compare the two turbines.####**  
  
####### unitlocation=='WTG01' ######  
  
wtg01=wtdata[wtdata$unitlocation=='WTG01',]  
s\_wtg01=wtg01$windspeed  
p\_wtg01=wtg01$power  
da\_wtg01=data.frame(s\_wtg01,p\_wtg01)  
x\_wtg01=fitcurve(da\_wtg01)

## Warning in s/c: longer object length is not a multiple of shorter object  
## length

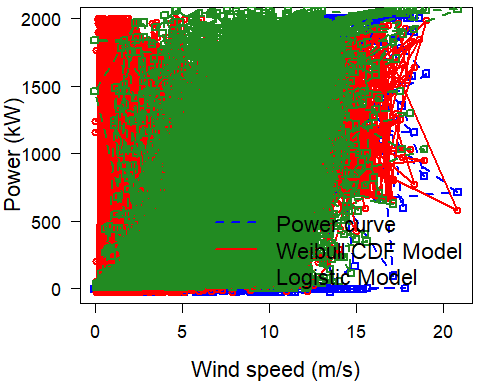
## Warning in (s/c)^k: longer object length is not a multiple of shorter  
## object length

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = -0.7642985 2.785122 1.168683 1.881557   
## Scale (C) = 5.411012 10.49532 12.78458 11.23951   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 2071.016   
## phi 2 = 9.01709   
## phi 3 = 1.692873   
## ===================================

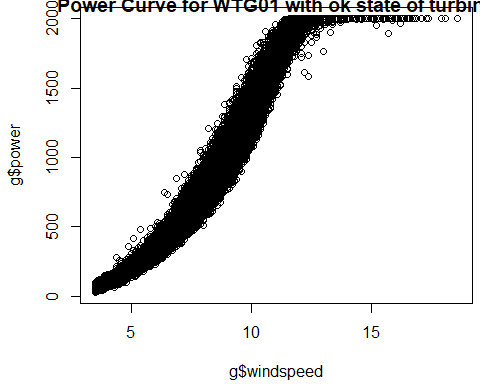
validate.curve(x\_wtg01)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 585.9004215 409.5677725  
## 2 MAE 430.0316836 266.6891278  
## 3 MAPE 63.4009831 68.3233665  
## 4 R2 0.1352186 0.5774187  
## 5 COR 0.4477855 0.7809602

plot(x\_wtg01)



g=wtg01 %>%filter(wtg\_state=='ok')%>%select(windspeed,power,wtg\_state)  
plot(g$windspeed,g$power,main='Power Curve for WTG01 with ok state of turbine')



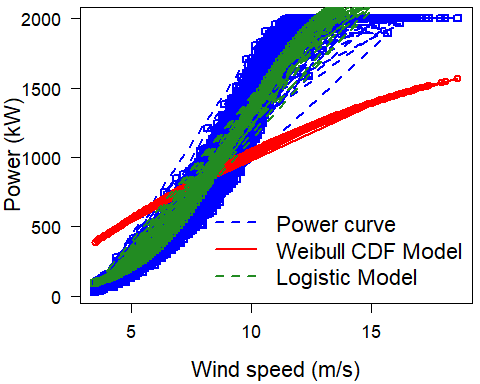
s\_g=g$windspeed  
p\_g=g$power  
da\_g=data.frame(s\_g,p\_g)  
x\_g=fitcurve(da\_g)

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = 1.206987   
## Scale (C) = 13.11049   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 2316.159   
## phi 2 = 9.349537   
## phi 3 = 1.872852   
## ===================================

validate.curve(x\_g)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 352.0040585 102.3320534  
## 2 MAE 293.8088105 79.9153721  
## 3 MAPE 36.1221033 11.0509363  
## 4 R2 0.5930491 0.9656070  
## 5 COR 0.9581362 0.9826626

plot(x\_g)



s\_wtg011=wtg01$ws\_norm  
p\_wtg011=wtg01$power\_norm  
da\_wtg011=data.frame(s\_wtg011,p\_wtg011)  
x\_wtg011=fitcurve(da\_wtg011)

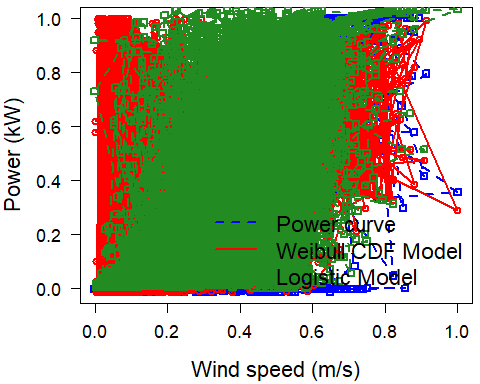
## Warning in s/c: longer object length is not a multiple of shorter object  
## length  
  
## Warning in s/c: longer object length is not a multiple of shorter object  
## length

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = -0.7642985 2.785122 1.168683 1.881557   
## Scale (C) = 0.2601448 0.5045827 0.6146432 0.5403612   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 1.035301   
## phi 2 = 0.433514   
## phi 3 = 0.08138813   
## ===================================

validate.curve(x\_wtg011)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 0.2928916 0.2047429  
## 2 MAE 0.2149728 0.1333179  
## 3 MAPE 63.4009831 68.3233665  
## 4 R2 0.1352186 0.5774187  
## 5 COR 0.4477855 0.7809602

plot(x\_wtg011)



####### unitlocation=='WTG02' ######  
  
wtg02=wtdata[wtdata$unitlocation=='WTG02',]  
s\_wtg02=wtg02$windspeed  
p\_wtg02=wtg02$power  
da\_wtg02=data.frame(s\_wtg02,p\_wtg02)  
x\_wtg02=fitcurve(da\_wtg02)

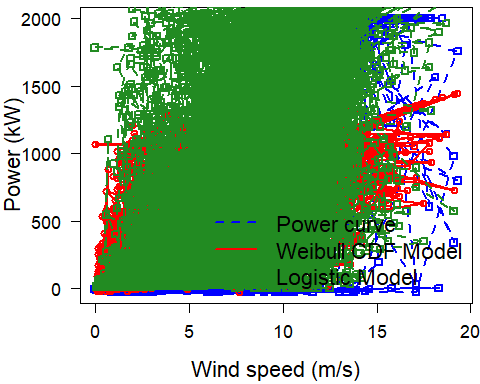
## Warning in s/c: longer object length is not a multiple of shorter object  
## length  
  
## Warning in s/c: longer object length is not a multiple of shorter object  
## length

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = 1.073908 1.073908   
## Scale (C) = 15.10679 15.10679   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 2196.57   
## phi 2 = 9.277109   
## phi 3 = 1.792731   
## ===================================

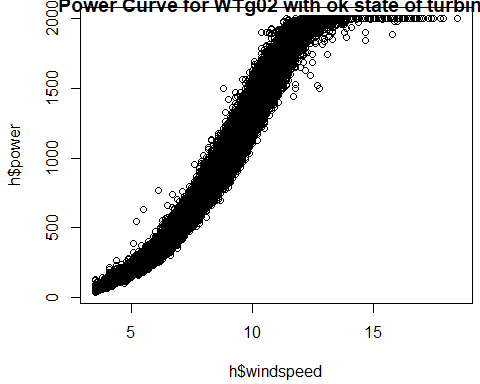
validate.curve(x\_wtg02)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 480.5288198 450.4981553  
## 2 MAE 384.0765294 287.0300683  
## 3 MAPE 60.7890180 78.5880074  
## 4 R2 0.4509192 0.5174044  
## 5 COR 0.7399664 0.7511340

plot(x\_wtg02)



h=wtg02 %>%filter(wtg\_state=='ok')%>%select(windspeed,power,wtg\_state)  
plot(h$windspeed,h$power,main='Power Curve for WTg02 with ok state of turbine')



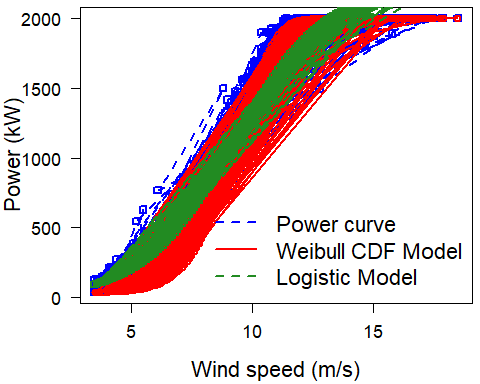
s\_h=h$windspeed  
p\_h=h$power  
da\_h=data.frame(s\_h,p\_h)  
x\_h=fitcurve(da\_h)

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = 7.550656   
## Scale (C) = 9.668968   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 2243.832   
## phi 2 = 9.276035   
## phi 3 = 1.853933   
## ===================================

validate.curve(x\_h)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 232.4971297 85.6989412  
## 2 MAE 199.4763089 66.8491705  
## 3 MAPE 101.3016833 8.9034805  
## 4 R2 0.8494042 0.9795389  
## 5 COR 0.9760796 0.9897202

plot(x\_h)



s\_wtg022=wtg02$ws\_norm  
p\_wtg022=wtg02$power\_norm  
da\_wtg022=data.frame(s\_wtg022,p\_wtg022)  
x\_wtg022=fitcurve(da\_wtg022)

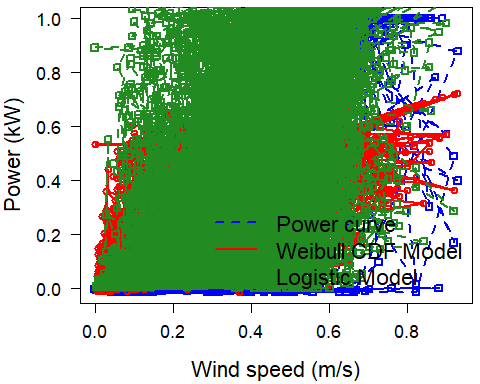
## Warning in s/c: longer object length is not a multiple of shorter object  
## length  
  
## Warning in s/c: longer object length is not a multiple of shorter object  
## length

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = 1.073908 1.073908   
## Scale (C) = 0.7262879 0.7262879   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 1.098065   
## phi 2 = 0.4460149   
## phi 3 = 0.08618901   
## ===================================

validate.curve(x\_wtg022)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 0.2402164 0.2252040  
## 2 MAE 0.1919999 0.1434863  
## 3 MAPE 60.7890180 78.5880074  
## 4 R2 0.4509192 0.5174044  
## 5 COR 0.7399664 0.7511340

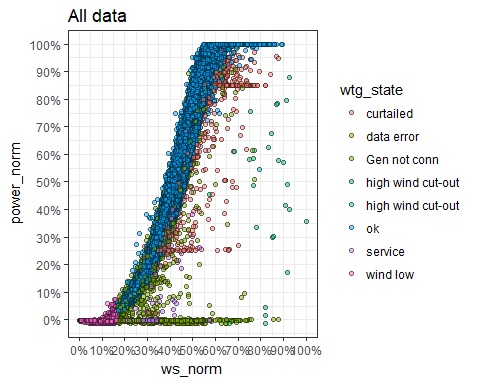
plot(x\_wtg022)



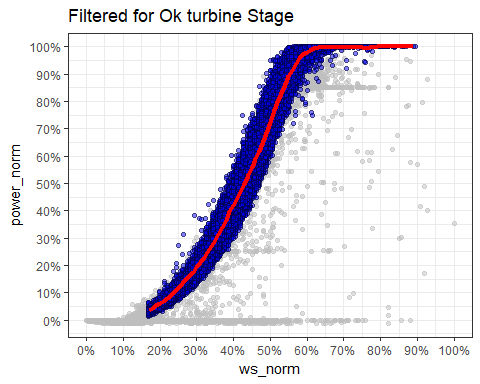
####With Normalized Data  
#### From the boxplot it is clear that Avg. wind speed is desceasing from June to October and then started increasing from October to January.  
  
#libraries  
library(MASS)  
library(tidyverse)  
library(knitr)  
library(viridis)  
library(dplyr)  
# binwise power curve function which we'll use a few times  
calc\_binwise\_pc<-function(df){  
 df %>%  
 mutate(ws\_bin=factor(round(ws\_norm,2))) %>%  
 group\_by(ws\_bin) %>%  
 summarise(n=n(),power\_norm=mean(power\_norm)) %>%  
 mutate(ws\_bin\_real=as.numeric(as.character(ws\_bin)))  
}  
  
#tidy  
  
#applying filtering techniques across two turbine with different capacities to normalize the wind speed and power measurements.  
wtdata<-mutate\_if(wtdata,is.character,as.factor)  
  
# normalise wind speed and power  
#wtdata<-mutate(wtdata,ws\_norm=windspeed/max(windspeed),power\_norm=power/max(power))  
  
# create container for power curves  
pc<-list()  
# Status code filtering   
# Understanding what are the Wind turbine States  
state.count<-wtdata %>%  
 group\_by(wtg\_state) %>%  
 summarise(n=n(),isProducing=sum(power\_norm>0)/n) %>%  
 arrange(desc(isProducing))  
  
kable(state.count)

|  |  |  |
| --- | --- | --- |
| wtg\_state | n | isProducing |
| high wind cut-out | 1 | 1.0000000 |
| ok | 44020 | 1.0000000 |
| curtailed | 4187 | 0.9949845 |
| high wind cut-out | 28 | 0.8928571 |
| Gen not conn | 3896 | 0.7271561 |
| wind low | 8658 | 0.2144837 |
| service | 473 | 0.1162791 |
| data error | 3 | 0.0000000 |

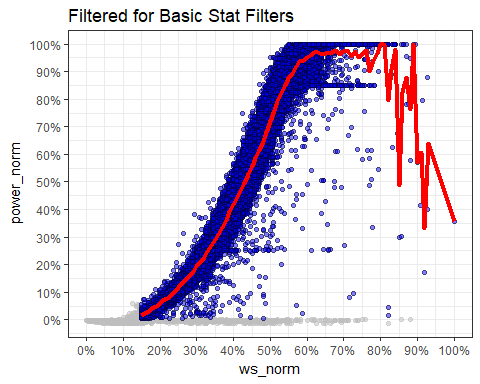
# Drawing the power curve  
ggplot(wtdata,aes(ws\_norm,power\_norm,fill=wtg\_state))+  
 geom\_point(color='black',pch=21,alpha=.5)+  
 theme\_bw()+  
 scale\_x\_continuous(breaks=seq(0,1,.1),labels=scales::percent)+  
 scale\_y\_continuous(breaks=seq(0,1,.1),labels=scales::percent)+  
 labs(title='All data')



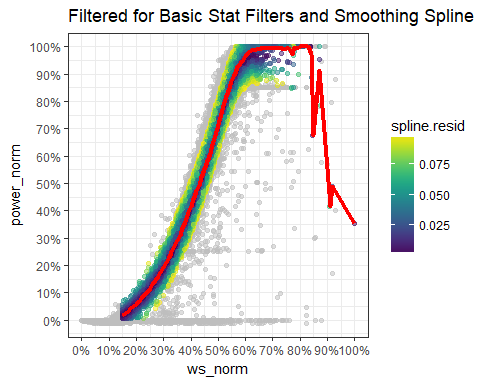
wtdata<-mutate(wtdata,filter.avail=wtg\_state=='ok')  
  
# STATUS CODE POWER CURVE  
# Filtering the power curve for status codes  
  
pc[['status']]<-wtdata %>%   
 filter(filter.avail & !is.na(ws\_norm) & !is.na(power\_norm)) %>%  
 calc\_binwise\_pc() %>%  
 mutate(method='status code')  
  
# show plot  
wtdata %>% filter(filter.avail) %>%  
 ggplot(aes(ws\_norm,power\_norm))+  
 geom\_point(data=filter(wtdata,!filter.avail),aes(ws\_norm,power\_norm),color='gray',alpha=.5)+  
 geom\_point(color='black',fill='blue',pch=21,alpha=.5)+  
 geom\_line(data=pc[['status']],aes(ws\_bin\_real,power\_norm),lwd=1.5,color='red')+  
 theme\_bw()+  
 scale\_x\_continuous(breaks=seq(0,1,.1),labels=scales::percent)+  
 scale\_y\_continuous(breaks=seq(0,1,.1),labels=scales::percent)+  
 labs(title='Filtered for Ok turbine Stage')



# STATISTICAL FILTERING  
# filter out obvious low/no ok periods  
  
wtdata <-mutate(wtdata,filter.basic\_stat=power>=10 & windspeed>3 & windspeed<25 )  
  
pc[['basic\_stat']]<-wtdata %>% filter(filter.basic\_stat) %>%  
 calc\_binwise\_pc() %>%  
 mutate(method='basic statistical')  
  
# show plot  
wtdata %>% filter(filter.basic\_stat) %>%  
 ggplot(aes(ws\_norm,power\_norm))+  
 geom\_point(data=filter(wtdata,!filter.basic\_stat),aes(ws\_norm,power\_norm),color='gray',alpha=.5)+  
 geom\_point(color='black',fill='blue',pch=21,alpha=.5)+  
 geom\_line(data=pc[['basic\_stat']],aes(ws\_bin\_real,power\_norm),lwd=1.5,color='red')+  
 theme\_bw()+  
 scale\_x\_continuous(breaks=seq(0,1,.1),labels=scales::percent)+  
 scale\_y\_continuous(breaks=seq(0,1,.1),labels=scales::percent)+  
 labs(title='Filtered for Basic Stat Filters')



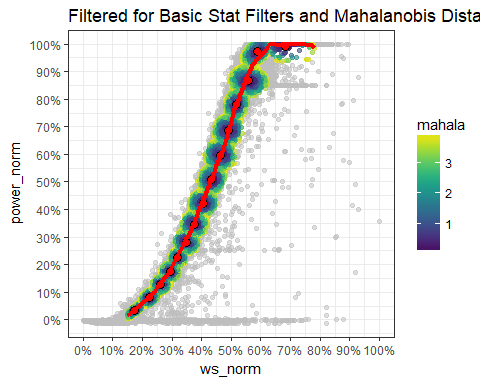
# Graph is not very good. Try to get rid of some of those outliers.  
  
## SMOOTHING SPLINE FILTERING  
# After applying the basic statistical filters, fit a smoothing spline to the data and define any point with a residual error >10% Power as an outlier.  
# create smoothing spline  
fit.spl<-wtdata %>%  
 filter(filter.basic\_stat) %>%  
 dplyr::select(ws\_norm,power\_norm) %>%  
 smooth.spline()  
  
# predict power using the fitted spline   
wtdata$power.spline<-predict(object=fit.spl,wtdata$ws\_norm)$y  
  
#define outliers  
wtdata<-mutate(wtdata,spline.resid=abs(power\_norm-power.spline),  
 filter.spline=spline.resid<.1)  
  
  
# calculate power curve  
pc[['spline']]<- wtdata %>%   
 filter(filter.basic\_stat & filter.spline) %>%  
 calc\_binwise\_pc()%>%  
 mutate(method='spline')  
  
# show plot  
wtdata %>% filter(filter.basic\_stat & filter.spline) %>%  
 ggplot(aes(ws\_norm,power\_norm,color=spline.resid))+  
 geom\_point(data=filter(wtdata,!filter.basic\_stat | !filter.spline),aes(ws\_norm,power\_norm),color='gray',alpha=.5)+  
 geom\_point(alpha=.5)+  
 scale\_color\_viridis()+  
 geom\_line(data=pc[['spline']],aes(ws\_bin\_real,power\_norm),color='red',lwd=1.5)+  
 theme\_bw()+  
 scale\_x\_continuous(breaks=seq(0,1,.1),labels=scales::percent)+  
 scale\_y\_continuous(breaks=seq(0,1,.1),labels=scales::percent)+  
 labs(title='Filtered for Basic Stat Filters and Smoothing Spline')



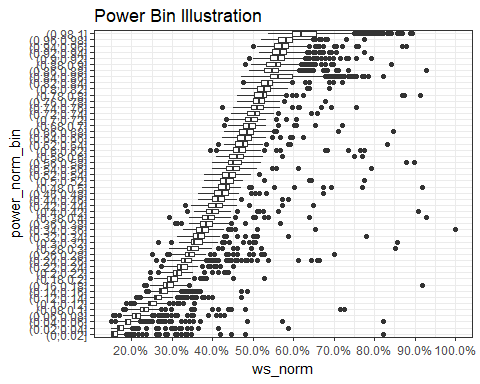
# MAHALANOBIS FILTERING  
# After applying the basic statistical filters, use a k-means approach to define centroids to the data,   
# then define any point where the mahalanobis distance from it's centroid is greater than a certain threshold.   
# As per standard 15 centroids seems to fit turbine power curves well when using all the data and outliers are defined where the mahalanobis distance is >2.5.   
# found that a mahalanobis threshold of 4.0 is a bit more conservative.  
  
# calculate centroids using k-means  
set.seed(1)  
k<- wtdata %>%  
 filter(filter.basic\_stat) %>%  
 dplyr::select(ws\_norm,power\_norm) %>%  
 kmeans(centers = 15,iter.max = 1000)

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 2608900)

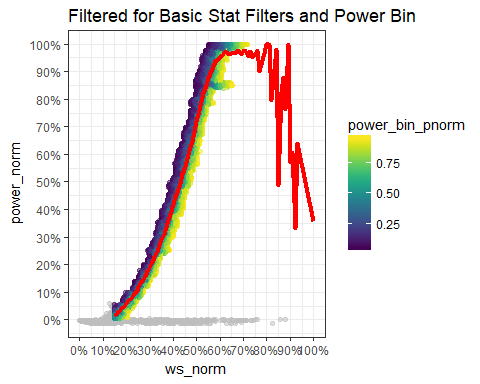
# format and append output  
wtdata<-mutate(wtdata,cluster=NA)  
wtdata$cluster[wtdata$filter.basic\_stat]<-k$cluster  
k\_centroids<-as.data.frame(k$centers)  
k\_centroids$cluster<-c(1:nrow(k\_centroids))  
names(k\_centroids)<-c('ws\_centroid','kw\_centroid','cluster')  
wtdata<-left\_join(wtdata,k\_centroids,by='cluster')  
  
  
# calculate the mahalanobis distance for points in each centroid  
wtdata<-mutate(wtdata,mahala=NA)  
for (i in 1:nrow(k\_centroids)){  
 x<-as.matrix(wtdata[wtdata$cluster==i & wtdata$filter.basic\_stat,c('ws\_norm','power\_norm')])  
 if(length(x)>10){  
 wtdata$mahala[wtdata$cluster==i & !is.na(wtdata$cluster)]<-  
 mahalanobis(x = x,center = c(k\_centroids$ws\_centroid[i],k\_centroids$kw\_centroid[i]),cov=cov(x))  
 }  
}  
  
# apply mahalanobis filter  
wtdata<-mutate(wtdata,filter.mahala=mahala<4)  
  
# calculate power curve  
pc[['mahala']]<-wtdata %>%   
 filter(filter.basic\_stat & filter.mahala) %>%  
 calc\_binwise\_pc()%>%  
 mutate(method='mahalanobis')  
  
# show plot  
wtdata %>% filter(filter.basic\_stat & filter.mahala) %>%  
 ggplot(aes(ws\_norm,power\_norm,color=mahala))+  
 geom\_point(data=filter(wtdata,!filter.basic\_stat),aes(ws\_norm,power\_norm),color='gray',alpha=.5)+  
 geom\_point(data=filter(wtdata,filter.basic\_stat | !filter.mahala),aes(ws\_norm,power\_norm),color='gray',alpha=.5)+  
 geom\_point(alpha=.5)+  
 geom\_point(data=k\_centroids,aes(ws\_centroid,kw\_centroid),size=3,color='black',fill='red',pch=21)+  
 scale\_color\_viridis()+  
 geom\_line(data=pc[['mahala']],aes(ws\_bin\_real,power\_norm),color='red',lwd=1.5)+  
 theme\_bw()+  
 scale\_x\_continuous(breaks=seq(0,1,.1),labels=scales::percent)+  
 scale\_y\_continuous(breaks=seq(0,1,.1),labels=scales::percent)+  
 labs(title='Filtered for Basic Stat Filters and Mahalanobis Distance')



# Power Binning  
#bin the data by power (2% bin of normalized power) and filter out points where wind speed > +/- 2 standard deviations)  
  
#create power bins  
wtdata<-mutate(wtdata,power\_norm\_bin=cut(power\_norm,seq(0,1,.02)))  
  
# calculate normal distribution of wind speed in each power bin  
power\_bins<- wtdata %>%  
 group\_by(power\_norm\_bin) %>%  
 summarise(power\_bin\_mean=mean(ws\_norm),  
 power\_bin\_sd=sd(ws\_norm))  
# filter data in each bin >2sd of the mean  
wtdata<-left\_join(wtdata,power\_bins,'power\_norm\_bin') %>%  
 mutate(power\_bin\_pnorm=pnorm(ws\_norm,power\_bin\_mean,power\_bin\_sd),  
 filter.powerBin=between(power\_bin\_pnorm,1-.9545,.9545))  
  
# show plot of bins  
wtdata %>%  
 filter(filter.basic\_stat) %>%  
 ggplot(aes(power\_norm\_bin,ws\_norm))+  
 geom\_boxplot()+  
 coord\_flip()+  
 theme\_bw()+  
 scale\_y\_continuous(breaks=seq(0,1,.1),labels=scales::percent)+  
 labs(title='Power Bin Illustration')



# calculate power curve  
pc[['powerBin']]<-wtdata %>%   
 filter(filter.basic\_stat) %>%  
 calc\_binwise\_pc()%>%  
 mutate(method='power bin')  
  
  
# show plot of filter  
wtdata %>% filter(filter.basic\_stat & filter.powerBin) %>%  
 ggplot(aes(ws\_norm,power\_norm,color=power\_bin\_pnorm))+  
 geom\_point(data=filter(wtdata,!filter.basic\_stat ),aes(ws\_norm,power\_norm),color='gray',alpha=.5)+  
 geom\_point(alpha=.5)+  
 scale\_color\_viridis()+  
 geom\_line(data=pc[['powerBin']],aes(ws\_bin\_real,power\_norm),color='red',lwd=1.5)+  
 theme\_bw()+  
 scale\_x\_continuous(breaks=seq(0,1,.1),labels=scales::percent)+  
 scale\_y\_continuous(breaks=seq(0,1,.1),labels=scales::percent)+  
 labs(title='Filtered for Basic Stat Filters and Power Bin')

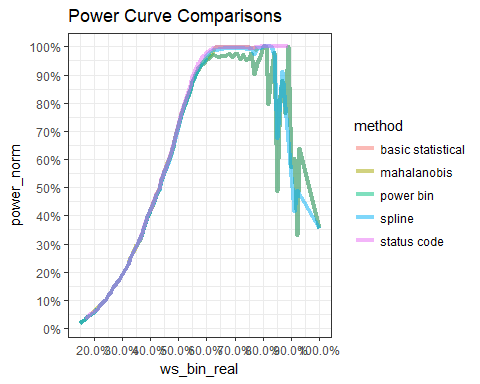


#POWER CURVES SIDE-BY-SIDE  
  
pc<-bind\_rows(pc)

## Warning in bind\_rows\_(x, .id): Unequal factor levels: coercing to character

## Warning in bind\_rows\_(x, .id): binding character and factor vector,  
## coercing into character vector  
  
## Warning in bind\_rows\_(x, .id): binding character and factor vector,  
## coercing into character vector  
  
## Warning in bind\_rows\_(x, .id): binding character and factor vector,  
## coercing into character vector  
  
## Warning in bind\_rows\_(x, .id): binding character and factor vector,  
## coercing into character vector  
  
## Warning in bind\_rows\_(x, .id): binding character and factor vector,  
## coercing into character vector

pc<-data.frame(pc)  
ggplot(pc,aes(ws\_bin\_real,power\_norm,color=method))+  
 geom\_line(lwd=1.5,alpha=.5)+  
 theme\_bw()+  
 scale\_x\_continuous(breaks=seq(0,1,.1),labels=scales::percent)+  
 scale\_y\_continuous(breaks=seq(0,1,.1),labels=scales::percent)+  
 labs(title='Power Curve Comparisons')



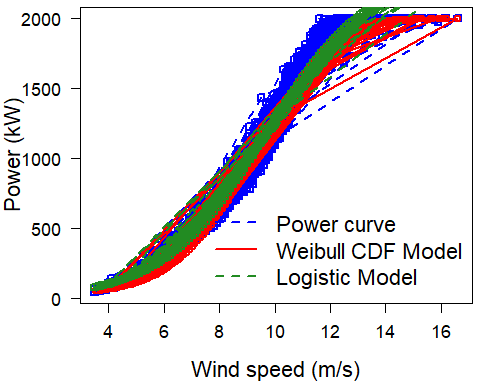
#### Converting into the data time format and storing it into the respective columns  
wtdata$ttimestamplocal=dmy\_hm(wtdata$ttimestamplocal)  
wtdata$month=month(wtdata$ttimestamplocal)  
wtdata$year=year(wtdata$ttimestamplocal)  
wtdata$day=day(wtdata$ttimestamplocal)  
wtdata$date=date(wtdata$ttimestamplocal)  
  
## filter the data month wise and wind turbine state='ok'  
dfjune = wtdata %>% filter(month==6,wtg\_state=='ok') %>% select(windspeed,power,month,wtg\_state)  
dfjuly = wtdata %>% filter(month==7,wtg\_state=='ok') %>% select(windspeed,power,month,wtg\_state)  
dfaug = wtdata %>% filter(month==8,wtg\_state=='ok') %>% select(windspeed,power,month,wtg\_state)  
dfsep = wtdata %>% filter(month==9,wtg\_state=='ok') %>% select(windspeed,power,month,wtg\_state)  
dfoct = wtdata %>% filter(month==10,wtg\_state=='ok') %>% select(windspeed,power,month,wtg\_state)  
dfnov = wtdata %>% filter(month==11,wtg\_state=='ok') %>% select(windspeed,power,month,wtg\_state)  
dfdec = wtdata %>% filter(month==12,wtg\_state=='ok') %>% select(windspeed,power,month,wtg\_state)  
  
## Analysis of Wind Curve month wise  
## June month  
  
s\_dfjune=dfjune$windspeed  
p\_dfjune=dfjune$power  
da\_dfjune=data.frame(s\_dfjune,p\_dfjune)  
x\_dfjune=fitcurve(da\_dfjune)

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = 4.755548   
## Scale (C) = 10.01027   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 2342.764   
## phi 2 = 9.664736   
## phi 3 = 1.852823   
## ===================================

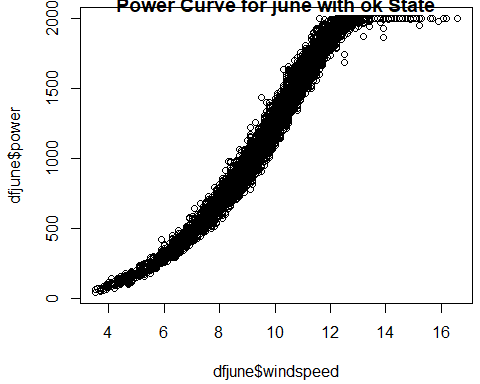
validate.curve(x\_dfjune)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 83.0961247 69.9377376  
## 2 MAE 69.6106916 54.1217032  
## 3 MAPE 11.2869148 5.5246234  
## 4 R2 0.9773145 0.9839302  
## 5 COR 0.9911493 0.9919530

plot(x\_dfjune)



plot(dfjune$windspeed,dfjune$power, main="Power Curve for june with ok State")



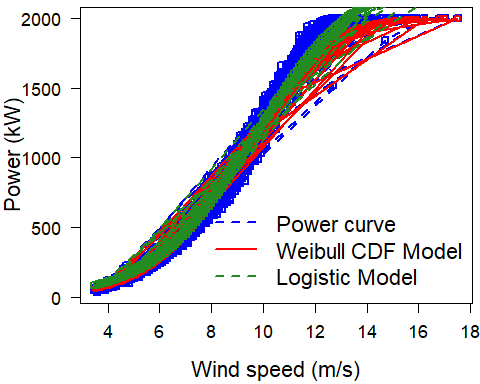
# For July Month  
s\_dfjuly=dfjuly$windspeed  
p\_dfjuly=dfjuly$power  
da\_dfjuly=data.frame(s\_dfjuly,p\_dfjuly)  
x\_dfjuly=fitcurve(da\_dfjuly)

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = 3.931009   
## Scale (C) = 10.04974   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 2316.866   
## phi 2 = 9.674186   
## phi 3 = 1.893599   
## ===================================

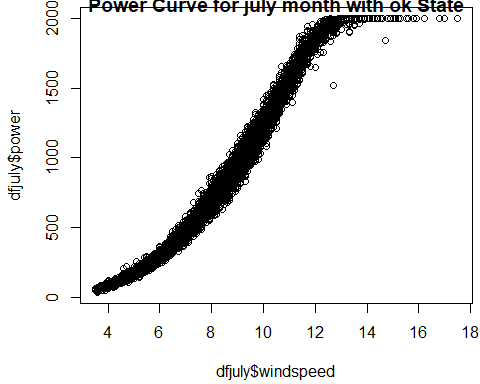
validate.curve(x\_dfjuly)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 63.9994688 56.4770530  
## 2 MAE 48.2740011 42.7356612  
## 3 MAPE 6.6949650 6.3885162  
## 4 R2 0.9859279 0.9890415  
## 5 COR 0.9933266 0.9945059

plot(x\_dfjuly)



plot(dfjuly$windspeed,dfjuly$power, main="Power Curve for july month with ok State")



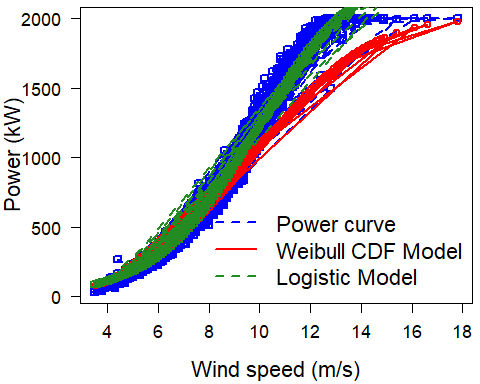
## For August Month  
s\_dfaug=dfaug$windspeed  
p\_dfaug=dfaug$power  
da\_dfaug=data.frame(s\_dfaug,p\_dfaug)  
x\_dfaug=fitcurve(da\_dfaug)

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = 3.175558   
## Scale (C) = 10.9262   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 2383.641   
## phi 2 = 9.839721   
## phi 3 = 1.958011   
## ===================================

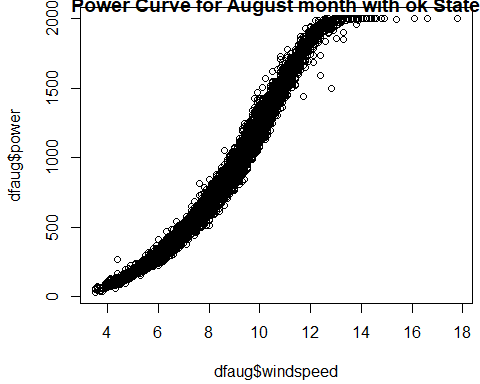
validate.curve(x\_dfaug)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 143.0745763 57.5528398  
## 2 MAE 98.9478318 43.6290774  
## 3 MAPE 11.0686433 6.1042369  
## 4 R2 0.9108819 0.9855797  
## 5 COR 0.9912189 0.9927641

plot(x\_dfaug)



plot(dfaug$windspeed,dfaug$power, main="Power Curve for August month with ok State")



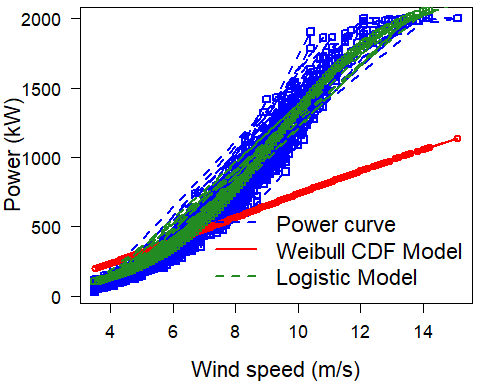
# For Sep month  
s\_dfsep=dfsep$windspeed  
p\_dfsep=dfsep$power  
da\_dfsep=data.frame(s\_dfsep,p\_dfsep)  
x\_dfsep=fitcurve(da\_dfsep)

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = 1.497477   
## Scale (C) = 17.13552   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 2211.102   
## phi 2 = 9.188975   
## phi 3 = 1.903093   
## ===================================

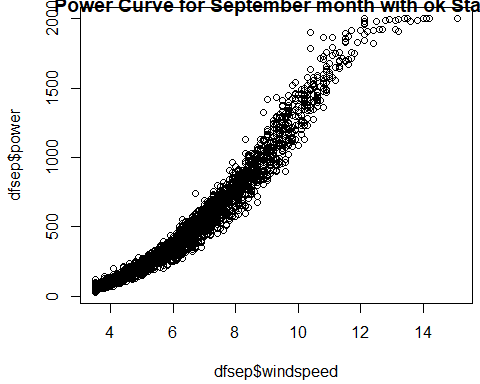
validate.curve(x\_dfsep)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 210.4678524 54.2006919  
## 2 MAE 136.5201589 37.0675967  
## 3 MAPE 31.5418093 10.0985082  
## 4 R2 0.6285014 0.9753626  
## 5 COR 0.9718781 0.9876593

plot(x\_dfsep)



plot(dfsep$windspeed,dfsep$power, main="Power Curve for September month with ok State")



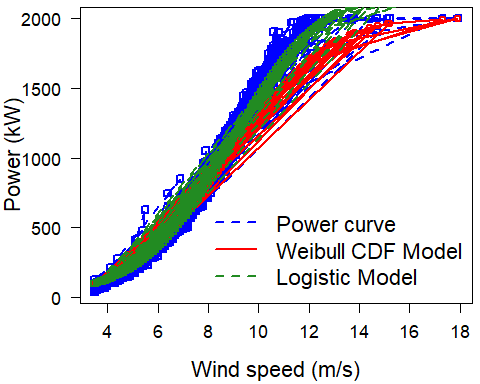
# For october Month  
s\_dfoct=dfoct$windspeed  
p\_dfoct=dfoct$power  
da\_dfoct=data.frame(s\_dfoct,p\_dfoct)  
x\_dfoct=fitcurve(da\_dfoct)

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = 3.288442   
## Scale (C) = 9.882469   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 2231.938   
## phi 2 = 8.914355   
## phi 3 = 1.780752   
## ===================================

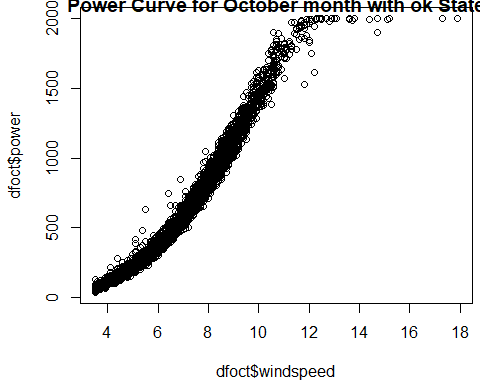
validate.curve(x\_dfoct)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 68.6309831 46.4110556  
## 2 MAE 44.3918030 33.6223987  
## 3 MAPE 8.1905198 7.0420264  
## 4 R2 0.9712989 0.9868750  
## 5 COR 0.9926321 0.9934245

plot(x\_dfoct)



plot(dfoct$windspeed,dfoct$power, main="Power Curve for October month with ok State")



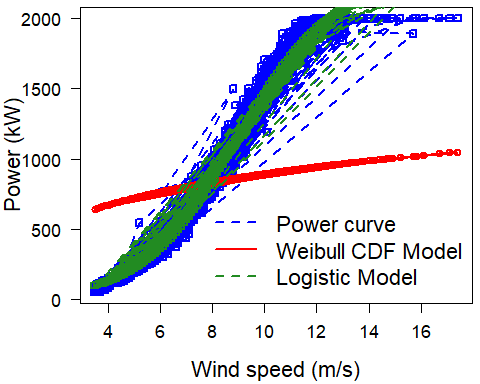
# For November month  
  
s\_dfnov=dfnov$windspeed  
p\_dfnov=dfnov$power  
da\_dfnov=data.frame(s\_dfnov,p\_dfnov)  
x\_dfnov=fitcurve(da\_dfnov)

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = 0.4250974   
## Scale (C) = 37.89628   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 2278.768   
## phi 2 = 8.944602   
## phi 3 = 1.77267   
## ===================================

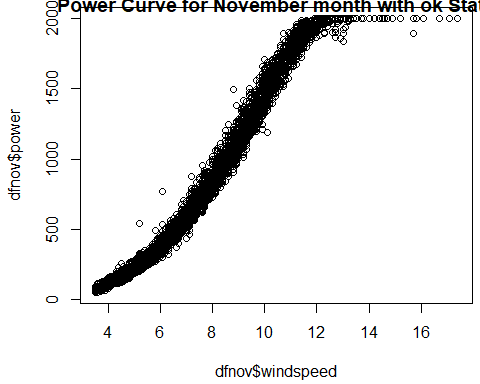
validate.curve(x\_dfnov)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 485.1739001 56.3180157  
## 2 MAE 407.1957264 41.2457303  
## 3 MAPE 49.3048517 6.1974175  
## 4 R2 0.2348225 0.9896899  
## 5 COR 0.9611049 0.9948319

plot(x\_dfnov)



plot(dfnov$windspeed,dfnov$power, main="Power Curve for November month with ok State")



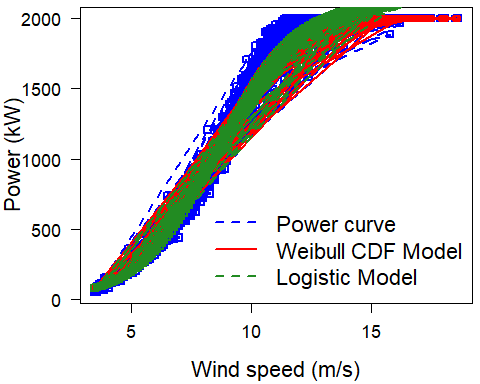
# For December month  
s\_dfdec=dfdec$windspeed  
p\_dfdec=dfdec$power  
da\_dfdec=data.frame(s\_dfdec,p\_dfdec)  
x\_dfdec=fitcurve(da\_dfdec)

## Weibull CDF model  
## -----------------  
## P = 1 - exp[-(S/C)^k]  
## where P -> Power and S -> Speed   
##   
## Shape (k) = 4.066103   
## Scale (C) = 9.188243   
## ===================================  
##   
## Logistic Function model  
## -----------------------  
## P = phi1/(1+exp((phi2-S)/phi3))  
## where P -> Power and S -> Speed   
##   
## phi 1 = 2125.975   
## phi 2 = 8.612468   
## phi 3 = 1.576264   
## ===================================

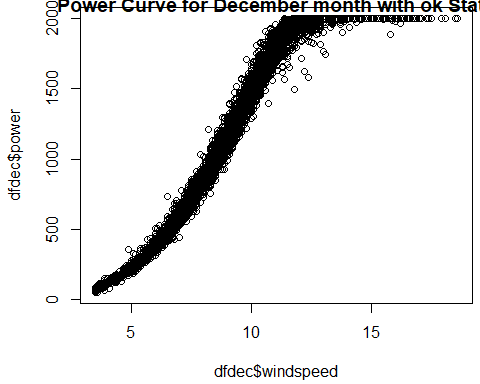
validate.curve(x\_dfdec)

## Metrics Weibull CDF Logistic Function  
## 1 RMSE 67.6203629 66.0065852  
## 2 MAE 51.6491235 52.4562348  
## 3 MAPE 4.8690243 5.2487126  
## 4 R2 0.9875842 0.9881698  
## 5 COR 0.9941451 0.9941209

plot(x\_dfdec)



plot(dfdec$windspeed,dfdec$power, main="Power Curve for December month with ok State")

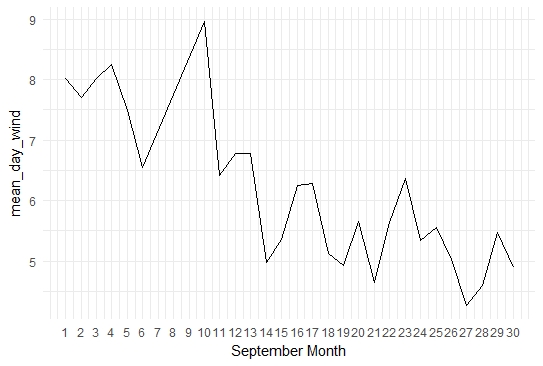


#### Further digging out the September Month as it is giving very poor MAPE Value for both wind Turbine

# For Wind Turbine 1

sept1=wtdata %>% filter(month==9,wtg\_state=='ok',unitlocation=='WTG01') %>% select(month,day,windspeed)%>% group\_by(day)%>% summarize(mean\_day\_wind=mean(windspeed))

ggplot(sept1,aes(day,mean\_day\_wind))+geom\_line()+theme\_minimal()+labs(x='day')+labs(x='September Month')+scale\_x\_continuous(breaks=seq(1,30,1))+scale\_y\_continuous(breaks=seq(1,12,1))+title(main = "In September Month Wind Turbine 1 with Ok state")



# Sep 14th,21st and 27th having the lowest windspeed in Wind Turbine 1 with Sep month and Ok State of Wind Turbine.

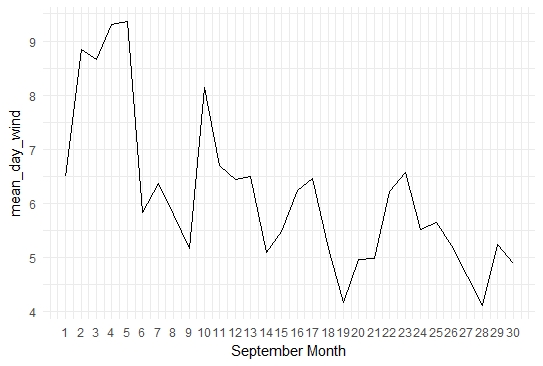
###################

# For wind turbine 2

sept2=wtdata %>% filter(month==9,wtg\_state=='ok',unitlocation=='WTG02') %>% select(month,day,windspeed)%>% group\_by(day)%>% summarize(mean\_day\_wind=mean(windspeed))

ggplot(sept2,aes(day,mean\_day\_wind))+

geom\_line()+theme\_minimal()+labs(x='day')+labs(x='September Month')+scale\_x\_continuous(breaks=seq(1,30,1))+scale\_y\_continuous(breaks=seq(1,12,1))+title(main = "In September Month Wind Turbine 2 with Ok state")



# In September month 9th,14th,19th,28th having the low value and data dips sudden on these dates.