NATIONAL RENEWABLE ENERGY LABORATORY

# A Guide to Using the WIND Toolkit Validation Code

W. Lieberman-Cribbin
*Colgate University*

C. Draxl and A. Clifton
*National Renewable Energy Laboratory*

**Technical Report**
NREL/TP-5000-62595
December 2014

Contract No. DE-AC36-08GO28308

# A Guide to Using the WIND Toolkit Validation Code

W. Lieberman-Cribbin
*Colgate University*

C. Draxl and A. Clifton
*National Renewable Energy Laboratory*

Prepared under Task No. WE11.0705

National Renewable Energy Laboratory
15013 Denver West Parkway
Golden, CO 80401
303-275-3000 • www.nrel.gov

**Technical Report**
NREL/TP-5000-62595
December 2014

Contract No. DE-AC36-08GO28308

*Cover Photos: (left to right) photo by Pat Corkery, NREL 16416, photo from SunEdison, NREL 17423, photo by Pat Corkery, NREL 16560, photo by Dennis Schroeder, NREL 17613, photo by Dean Armstrong, NREL 17436, photo by Pat Corkery, NREL 17721.*

NREL prints on paper that contains recycled content.

## Executive Summary

In response to the U.S. Department of Energy's (DOE's) goal of using 20% wind energy by 2030, the Wind Integration National Dataset (WIND) Toolkit was created to provide information on wind speed, wind direction, temperature, surface air pressure, and air density on more than 126,000 locations across the United States from 2007 to 2013. The numerical weather prediction model output, which is gridded at 2 kilometers and at a 5-minute resolution, was further converted to detail the wind power production time series of existing and potential wind facility sites.

Users of the WIND toolkit need to know if the the information presented in the WIND Toolkit is accurate and that errors are known, so corrective steps can be taken. Therefore, a team of researchers from the National Renewable Energy Laboratory are providing a validation code written in R that will be made public to allow users to validate data at their own locations. Validation is based on statistical analyses of wind speed, using error metrics such as bias, root mean squared error, centered root mean squared error, mean absolute error, and percent error. Additionally, plots of diurnal cycles, annual cycles, wind roses, histograms of wind speed, and quantile-quantile plots are created to visualize how well observational data compares to model data. Ideally, validation will confirm beneficial locations to utilize wind energy and encourage regional wind integration studies using the WIND Toolkit.

## Acknowledgments

This work was supported by the U.S. Department of Energy under Contract No. DE-AC36-08GO28308 with the National Renewable Energy Laboratory. Funding for the work was provided by the DOE Office of Energy Efficiency and Renewable Energy, Wind and Water Power Technologies Office. Wil Lieberman-Cribbin would like to thank Ms. Lung and Mr. Giron for their guidance throughout his Science Undergraduate Laboratory Internship.

## Table of Contents

## List of Figures

## List of Tables

## 1. Introduction

With wind as a promising renewable energy resource in the United States, there is a need for more detailed data that can be used in wind integration studies [1, 2, 3]. In response to this need, the National Renewable Energy Laboratory (NREL) has produced a national wind resource dataset based on a numerical weather prediction model [1].

This paper focuses on the Wind Integration National Dataset (WIND) Toolkit, a wind resource dataset produced with the National Center for Atmospheric Research's Weather Research and Forecasting (WRF) model [1, 4, 5, 6]. In essence, the WIND Toolkit provides modeled data about wind speed, wind direction, and wind power production for the contiguous United States for 2007 to 2013 on more than 116,000 land-based and 10,000 offshore locations [1]. As such, it will facilitate grid integration research and wind resource assessments and promote deployment.

For dataset users, it is important that the information presented in the WIND Toolkit is accurate and that errors are known, so corrective steps can be taken. To determine the efficacy of the WIND Toolkit, observational data can be used to validate the model at individual locations.

Wind speed and wind direction data in the WIND Toolkit is presented at 100 meters (m) and can be downloaded at http://maps.nrel.gov/wind_prospector. Observational data used in validation varies among locations and has to be provided by the user. A basic quality control is completed on both observational and model data, using thresholds to provide a cutoff of wind speed and wind direction values. Values below 0 m/second (m/s) or above 40 m/s for wind speed were excluded. Data below 0 degrees or above 360 degrees for wind direction were also removed from analyses. Users are responsible for controlling the quality of observational data and ensuring that data is available at 100 m. No corrections were made in the validation to account for data at heights other than 100 meters. An example of validation was performed at the National Wind Technology Center (NWTC) at NREL.

Validation relies on observational and model data and uses statistical analyses to produce error metrics (Table 1). These metrics test how information presented in the WIND Toolkit compares to observational data at individual locations. Ideally, validation will confirm the accuracy of the WIND Toolkit and depict optimal regions to capture wind energy; however, the validation code can be used to calculate error metrics between any two datasets.

The validation code is written in R version 3.1. R is a language and environment for statistical computing and graphics, and is available for free. It is optimized to be straightforward for the user. Given the applicability and usability of the validation code, the WIND Toolkit can be used in future wind integration studies and to promote wind energy in the United States.

**Table 1. Error Metrics Used in the Validation Process**

| Metric | Description |
|---|---|
| Root-mean-squared-error (RMSE) | $RMSE^2 = \frac{1}{N_p}\sum_{k=0}^{N_p}(F_k - O_k)^2 = CRMSE^2 + BIAS^2$; $N_p$ is the number of available forecast ($F$) – observation ($O$) pairs. The RMSE can be split into the systematic and random components (bias and centered-root-mean-squared-error) of the RMSE [7]. |
| BIAS (bias) | $BIAS = \bar{F} - \bar{O}$; $\bar{F}$ and $\bar{O}$ are the forecast and observation averages over $N_p$ values. The bias is the systematic component of the error and describes the differences in the mean of two time series. |
| Centered-root-mean-squared-error (CRMSE) | $CRMSE = \sqrt{\frac{1}{N_p}\sum_{k=1}^{N_p}[(F_k - \bar{F}) - (O_k - \bar{O})]^2}$; The CRMSE is the random component of the error, and describes the centered pattern of the error. In other words, the differences in wind speed variations around the mean. |
| Mean absolute error (MAE) | $MAE = \frac{1}{N_p}\sum_{k=1}^{N_p}\lvert F_k - O_k\rvert$. The MAE is a mean of the absolute errors. |
| Percentage error | The percentage error is the mean of the absolute errors in percent of the observed wind speed. |

## Using This Document

This document is intended for users with varying skill levels and familiarity with R. The authors recommend that newcomers to R read Section 2, which includes help resources and advantages to using the software as well as details on how to install R/R Studio and a text viewer.

All users are encouraged to read Section 3 because it provides an overview of a configuration file (3.1) and the components of a configuration file (3.2). Additionally, all users are strongly encouraged to read Section 3.3, which explains the lines in the code the user must change to define a data root directory. Also, Section 3.4 illustrates how to create and store appropriate folders for "data" and "figures." Reviewing Section 3.5 is equally important as it explains how to define a coding directory and source functions. Furthermore, the summary (Section 3.6) reiterates the main steps of the analysis.

The authors also recommend all users review Section 4, which describes the functions used in the validation code. For a basic R user, it is not essential to understand everything about these functions, but having a general idea of what they do is beneficial. Additionally, more in-depth descriptions of the functions can be found in the appendix, along with example outputs of functions, figures, and tables for the NWTC.

Users are encouraged to review the "How to Use Example Data to Perform a Validation" and "Adapting Examples to Validate User Data" sections in the appendix. The former section describes how to perform an example validation using the files in the attached zip folder. The latter section details how to adapt example data to perform additional validations.

It is possible to alter the code to perform additional analyses beyond what our validation provides; however, users are encouraged to be careful doing so, as many outputs and data frames are relative to each other.

# 2. Preparation

## 2.1 Installing R/R Studio

To complete the validation, R must be installed. R can be downloaded at http://cran.rstudio.com/.

In addition, we suggest using R Studio, an updated computer graphical user interface for R that can be downloaded at http://www.rstudio.com/products/rstudio/download/.

## 2.2 Why use R Studio?

R Studio is a free, prevalent, and open-source software. Work done in R Studio can be easily shared and includes excellent help resources. Popular sources of help can be found at:

- http://stackoverflow.com/questions/tagged/r
- http://cran.r-project.org/doc/FAQ/R-FAQ.html
- http://www.cookbook-r.com/
- http://www.statmethods.net/interface/help.html.

## 2.3 Using a Text Viewer

Users are encouraged to use a text viewer to create a configuration file and view files, such as Notepad++, which can be downloaded at http://notepad-plus-plus.org/download/v6.6.7.html.

# 3. Required Steps for Analysis

## 3.1 Overview of a Configuration File

A configuration file is a .txt file created by the user that contains all the necessary information used in the validation. We suggest users to review the layout of the configuration file in the zip folder. It is imperative that the header of the configuration file used in validation be the same as the header in the example configuration file because the validation code will not work. For this reason, users are encouraged to copy and paste the header of the attached configuration file into their configuration file.

To perform a validation, a configuration file needs to contain information about model and observational data for the same site. More specifically, the configuration file specifies where information is located in the data (for example, wind speed data is located in column 3).

The minimal required information present in the configuration file must be the site name, data file, data type, and location of wind speed (i.e., the column in the data file that contains wind speed data: “WS.col”). Additionally, the location of the time stamp (“time.stamp.col”) or the date column (“date.col”) and the time column (“time.col”) must be specified. Furthermore, the time stamp format (i.e., the “time.stamp.format” column) or date (“date.format” column) and time formats (“time.format” column) must be specified for observational and model data. Example entries in these columns are discussed in Sections 3.2.2 and 3.2.5.

The rationale behind specifying the time stamp location or date and time location is because these entries are mutually exclusive. In other words, there is no need to have separate entries for date and time columns if the date and time are in the same column (“time stamp” column). Thus, if users have an entry in the “time.stamp.col,” they must have an entry in the “time.stamp.format” column. Similarly, if users have entries in the date.col and time.col, they must have entries in the “date.format” and “time.format” columns.

If users enter the location of wind speed and wind direction data in the configuration file, the validation code will create wind roses. Similarly, if users specify wind speed and/or wind direction and/or temperature information, the code will generate time series of these variables.

Model data for specific locations can be downloaded from the WIND Toolkit at http://maps.nrel.gov/wind_prospector; however, observational data must be provided by the user.

## 3.2 Components of the Configuration File

In creating a configuration file (a simple text file), the first line (header) needs to be:

```
site.name    data.file    data.type    skip    sep    time.stamp.col time.stamp.format
        date.col     date.format    time.col     time.format    tz      WS.colWSms
        WD.col       WDdeg          Temp.col     Temp.C         Temp.K      Temp.F
```

Users should copy and paste this header into their new configuration file. If users wish to alter how the configuration file is read in the main script, a “tab” should be used to separate column headers. See the example entry for the NWTC (lines 13 and 14 of the configuration file).

### 3.2.2 Site.name, data.file, and data.type

Because the site we are validating is named NWTC, we enter NWTC under the "site.name" column (column 1 of the configuration file). The filename.txt of the data file is included under the "data.file" column (column 2), while the data type is either "Obs" (observations) or "Model" (column 3). If observational data is contained in multiple files, such as "data_obs_one.txt" and "data_obs_two.txt," they must be separate entries, or rows, in the configuration file. The validation code will recognize multiple entries for the same site name and combine these files to produce one output file for "data.obs."

### 3.2.3 Skip

The "skip" column is indicative of how many lines need to be "skipped," i.e., are header lines (column 4). In other words, if the first five lines of user data do not contain data users want to analyze, enter a "5" under the "skip" column to skip those lines.

### 3.2.4 Separation (sep)

The "sep" column refers to the separation method used in the data (column 5). If the data columns are separated by "tab," enter \t under the "sep" column in the configuration file. Note that if data are separated by a white space, do not enter a space in the configuration file—instead, enter two quotation marks ("").

### 3.2.5 Date and Time

If the date and time are located in the same column of your data, enter the column number under the "time.stamp.col" (column 6), and the format under "time.stamp.format" (column 7). An example where the date and time are in the same column is 20070423 0723. In this case, the format is %Y%m%d %H%M and the separation is "\t." If users are unfamiliar with time stamps, entering "?strptime" in R will display abbreviations for time stamp formats (Table 2).

**Table 2. Examples of Time Stamp Abbreviations**

| Abbreviation | Explanation |
|---|---|
| %Y | Full year (e.g., 2007) |
| %m | Month as a decimal (01–12) |
| %d | Day as a decimal number (01–31) |
| %H | Hour as a decimal number (00–23) |
| %M | Minute as a decimal number (00–59) |

If date and time information are in the same column, users must not have entries for "date.col" (column 8), "date.format" (column 9), "time.col" (column 10), and "time.format" (column 11). Skip over these columns by hitting "tab" in the configuration file.

If "date" and "time" information exists in separate columns, users must enter the appropriate information under the "date.col," "date.format," "time.col," and "time.format" columns. If this is the case, the validation code recognizes that date and time exist in separate columns and pastes them together. If date and time information exist in separate columns, users must skip over the “time.stamp.col” and “time.stamp.format” columns by pressing “tab.”

### 3.2.6 Time Zone (tz)

The “tz” column indicates that the time zone data was pulled from R (column 12). The validation code then converts this time zone to Coordinated Universal Time (UTC). Because NWTC data were available in Mountain Standard Time, “MST” was entered under the “tz” column. If data were pulled in Greenwich Mean Time, users would still need to enter “GMT” under the “tz” column. Although there will be no time zone conversion, the validation code requires a time zone in the configuration file.

### 3.2.7 Wind Speed (WS.col)

“WS.col” refers to the column that the wind speed data is located in (column 13). “WSms” is a TRUE/FALSE column indicating if the wind speed data is in m/s (column 14). If data is in miles per hour, users must indicate FALSE in the WSms column, and the validation code will convert wind speed data to m/s.

### 3.2.8 Wind Direction (WD.col)

Similarly, “WD.col” refers to the column wind direction data is located in the data file (column 15). “WDdeg” is a TRUE/FALSE column assessing if wind direction data was taken in degrees (column 16). If the data was taken in radians, the code converts the wind direction data to degrees.

### 3.2.9 Temperature

Temperature data are not required to run the validation code. If the user provides temperature data, the code constructs a time series of this data. If this is applicable, users must enter the appropriate column for temperature data under “Temp.col” (column 17). “Temp.C” (column 18), “Temp.K” (column 19), and “Temp.F” (column 20) are all logical and indicate the units data was taken in (Celsius, Kelvin, or Fahrenheit, respectively). If data are not in Celsius, temperature data are converted to Celsius.

## 3.3 Defining a Data Root Directory and the Name of the Configuration File

The user must also define a root directory in which the configuration file is located. The purpose of this directory is to load the configuration file. In Figure 1, the configuration file is located in a folder named CONFIG and is a .txt file. Line 91 of the main script loads the configuration file and line 92 specifies the name of the configuration file.

```
(91) data.root = "C:/Users/xxxxxxxxx/Desktop/CONFIG"

(92) config.file = "load.txt"
```

The user needs to change (91-92) of the main script.

Defining a data.root directory is also necessary when creating a location to store observational data (“data.obs”), model data (“data.model”), and figures for each site. In line 131, we refer to a “site directory” for the site we want to validate. This “site.dir” refers to a folder with the name of the site and is stored under the data root directory. In line 132, we refer to a folder named “data,” where “data.obs” and “data.model” are stored. In line 133, we refer to a folder named “figures,” where the figures will be stored for each site.

(131) site.dir = file.path(data.root,site)

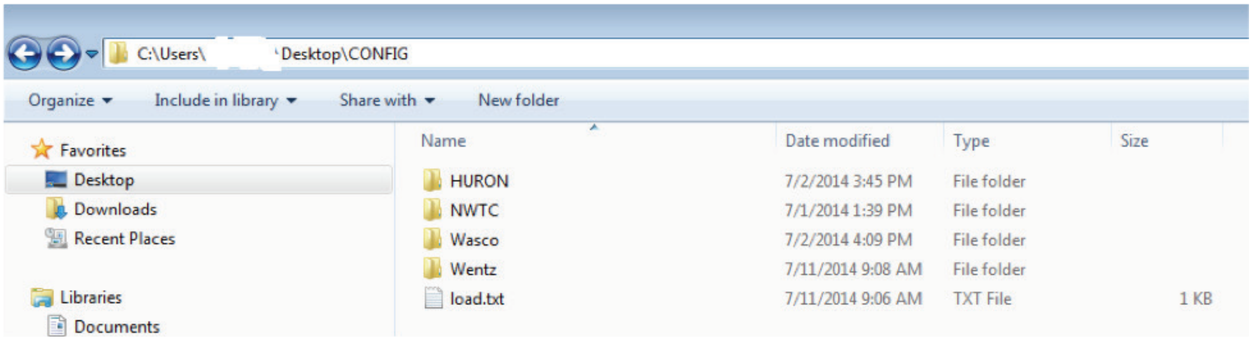
(132) data.dir = file.path(data.root,site,"data")

(133) figure.dir = file.path(site.dir,"figures")

DO NOT alter these lines in the code; this information is listed here to provide clarity.

## 3.4 Creating a Location Where Data and Figures Will Be Stored

**Important**: The user needs to create a folder with the name of the site located under the data.root directory. An example of the file structure is shown in Figure 1. If the data root directory is defined as "C:/Users/xxxxxxxxx/Desktop/CONFIG”, the configuration file, in this case called “load.txt,” is displayed, as well as folders for the names of the sites.

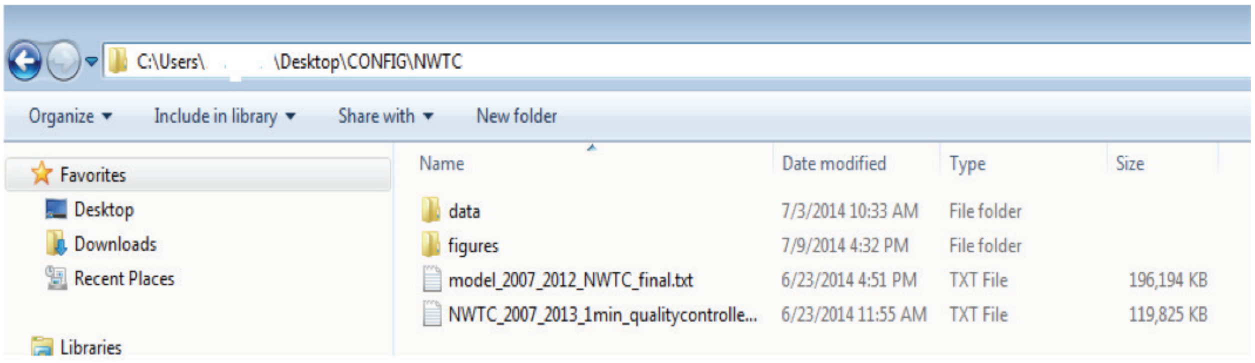


| Name | Date modified | Type | Size |
|---|---|---|---|
| HURON | 7/2/2014 3:45 PM | File folder | |
| NWTC | 7/1/2014 1:39 PM | File folder | |
| Wasco | 7/2/2014 4:09 PM | File folder | |
| Wentz | 7/11/2014 9:08 AM | File folder | |
| load.txt | 7/11/2014 9:06 AM | TXT File | 1 KB |

**Figure 1. Example location and contents of the data root directory**

Opening one of the site.name folders reveals two folders and two .txt files (Figure 2); however, if users have more than one file for observations or model data, they will have more than two files. The two (or possibly more) files contain observational and model data for that site. In this case, the model data is named “model_2007_2012_NWTC_final.txt” and the observational data is named “NWTC_2007_2013_1_min_qualitycontrolled.txt.”

The user has to create two folders under each site-name folder (Figure 2). These folders should be called “data” and “figures,” all in lowercase letters. There should be nothing in these folders originally. Thus, the observational and model data files need to be stored inside their respective folder name under the data.root directory, and not in the “data” folder. When Part 1 of the validation code runs, it will put data_obs and data_model in the “data” folder, and put plots and tables in the “figures” folder. Figure 3 shows example contents of the “data” and “figures” folders after validation.

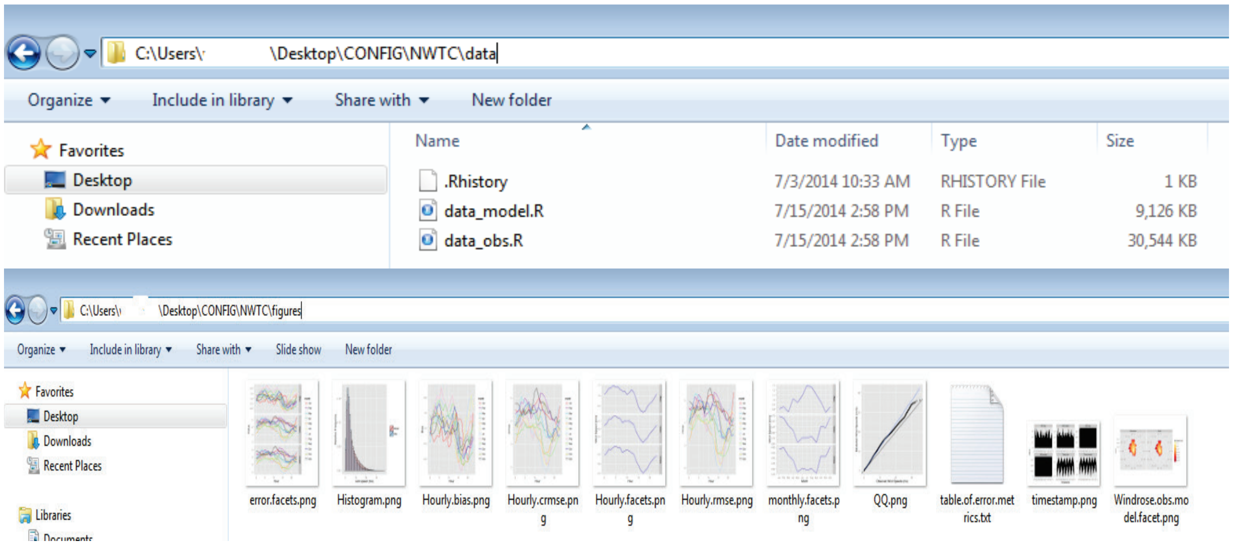

C:\Users\ \Desktop\CONFIG\NWTC

Organize ▾ Include in library ▾ Share with ▾ New folder

Favorites
Desktop
Downloads
Recent Places
Libraries

| Name | Date modified | Type | Size |
|---|---|---|---|
| data | 7/3/2014 10:33 AM | File folder | |
| figures | 7/9/2014 4:32 PM | File folder | |
| model_2007_2012_NWTC_final.txt | 6/23/2014 4:51 PM | TXT File | 196,194 KB |
| NWTC_2007_2013_1min_qualitycontrolle... | 6/23/2014 11:55 AM | TXT File | 119,825 KB |

**Figure 2. Location where data files need to be placed (under root directory)**


C:\Users\ \Desktop\CONFIG\NWTC\data

Organize ▾ Include in library ▾ Share with ▾ New folder

Favorites
Desktop
Downloads
Recent Places

| Name | Date modified | Type | Size |
|---|---|---|---|
| .Rhistory | 7/3/2014 10:33 AM | RHISTORY File | 1 KB |
| data_model.R | 7/15/2014 2:58 PM | R File | 9,126 KB |
| data_obs.R | 7/15/2014 2:58 PM | R File | 30,544 KB |

C:\Users\ \Desktop\CONFIG\NWTC\figures

Organize ▾ Include in library ▾ Share with ▾ Slide show New folder

Favorites
Desktop
Downloads
Recent Places
Libraries
Documents

error.facets.png
Histogram.png
Hourly.bias.png
Hourly.crmse.png
Hourly.facets.png
Hourly.rmse.png
monthly.facets.png
QQ.png
table.of.error.metrics.txt
timestamp.png
Windrose.obs.model.facet.png

**Figure 3. Example contents of "data" folder (top) and "figures" folder (bottom)**

## 3.5 Defining a Coding Directory

The user must alter the coding directory (line 96) that defines the location of functions used in the validation. In this example, the functions are located in a folder titled "functions" on the desktop. Line 97 of the main script loads these functions into R.

(96) code.dir = "C:/Users/xxxxxxxxxxxx/Desktop/functions/"

(97) code.files = dir(code.dir, pattern = "[.R]")

The user has to change line 96 of the main script.

## 3.6 Summary

To run the provided validation code, users must perform the following steps:

1. Create a configuration file and store it under the data root directory. Define the data root directory in line 91 and the name of the configuration file in line 92 of the main script.

2. Create a folder under the data root directory with the site name ("site.name") used in the configuration file (e.g., NWTC).
3. Under each site-name folder, store the model and observational data for that site. Users should not store the data they wish to load into R under the "data" folder.
4. Create two folders (will originally be empty) titled "data" and "figures."
5. Define the location of functions used in the validation (line 96 of the main script).

Figure 4 and Figure 5 provide flowcharts to better illustrate the entire validation process. Figure 4 illustrates Part 1 of the validation: loading data into R; and Figure 5 depicts Part 2 of the process: how error metrics are calculated and functions are used.

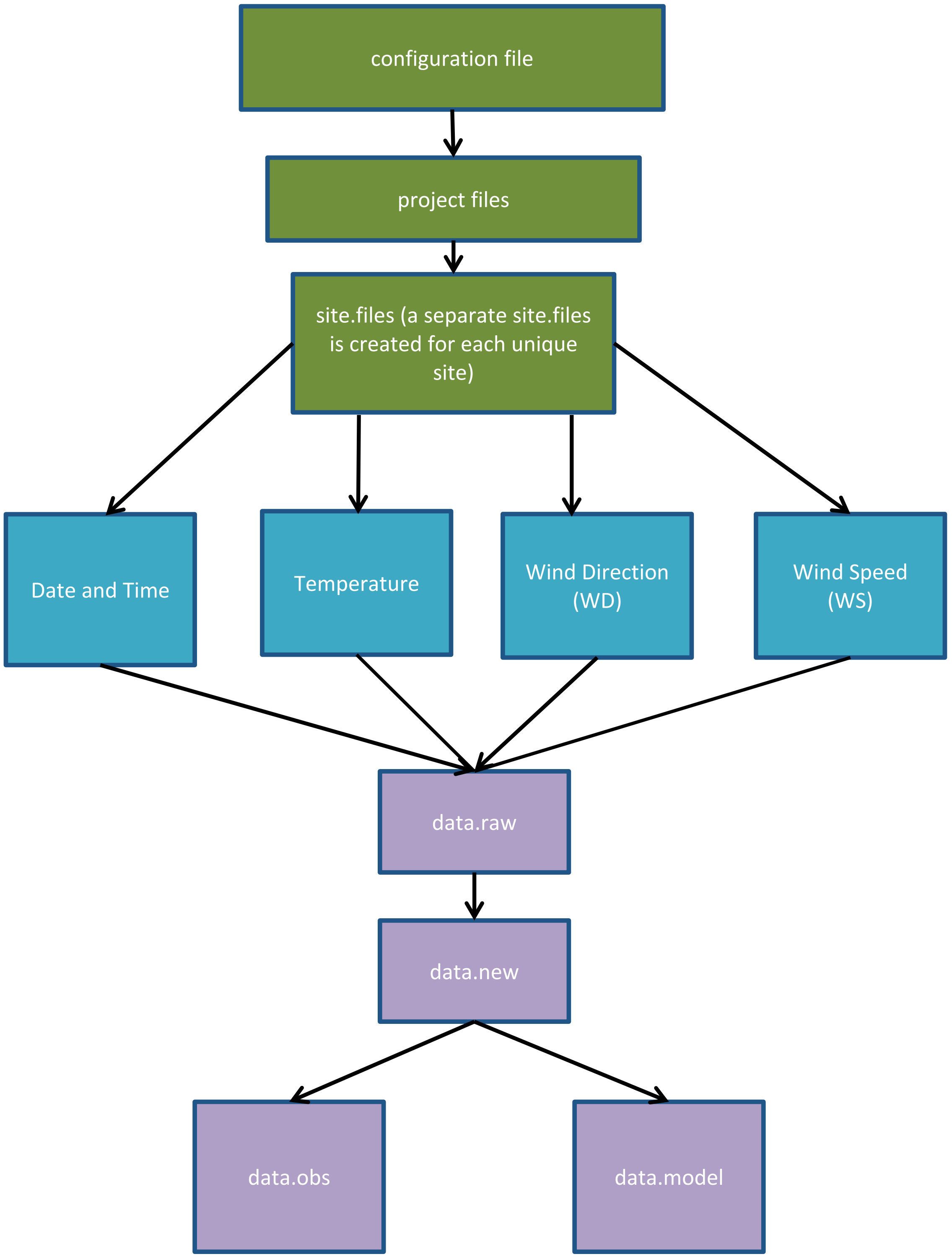


**Figure 4. Flowchart depicting the process of loading data into R (Part 1 of the code). After creating a configuration file and the appropriate directories, Part 1 of the validation returns "data.obs" and "data.model" for each site name. Green boxes represent files with input data, teal boxes represent information stored in input data, and purple boxes correspond to processed data in .R format.**

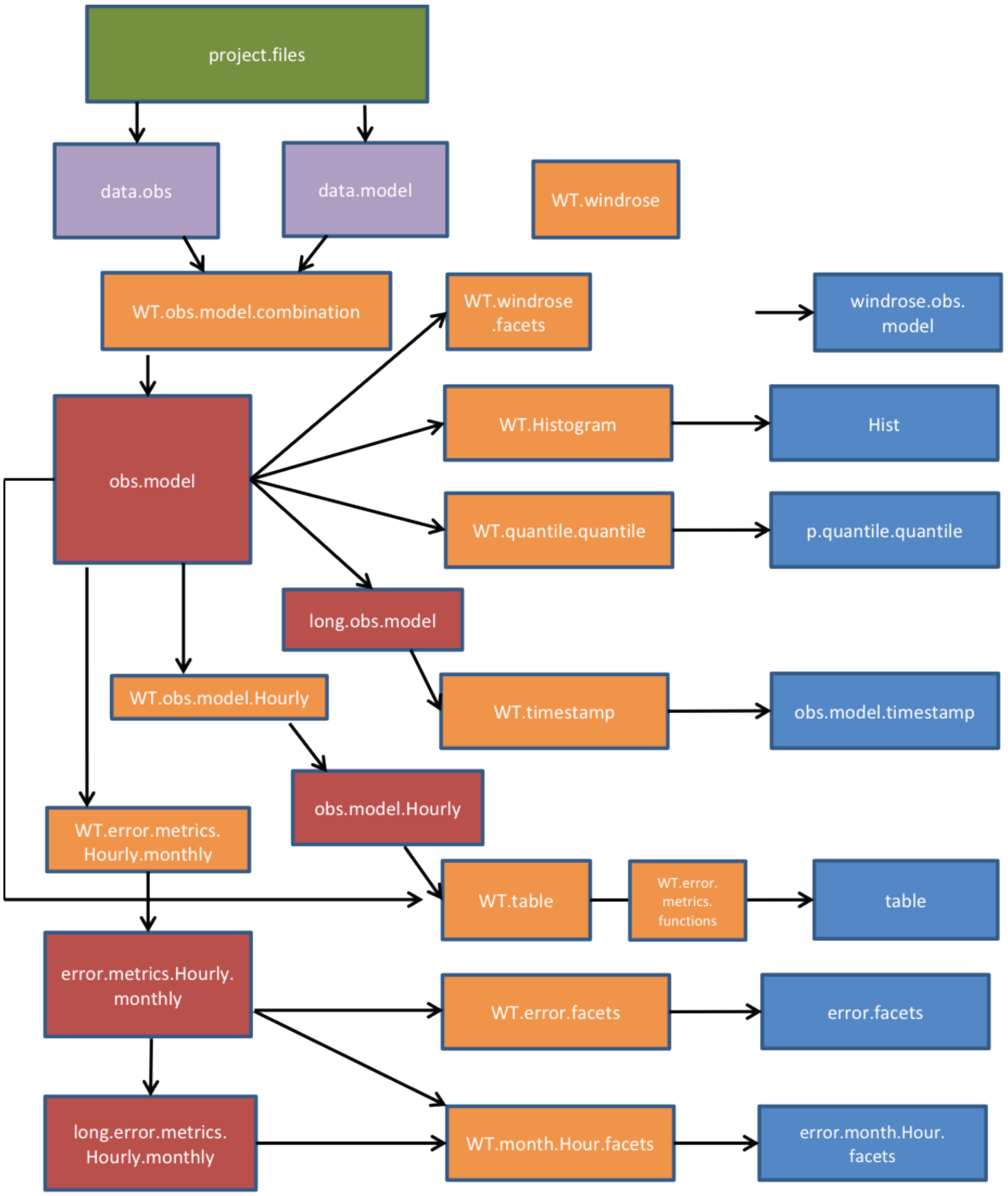


**Figure 5. Flowchart illustrating completing error metrics on data (Part 2 of the code). Error metrics are computed with data included in the project files. Red boxes represent created data frames, orange boxes are functions, and blue boxes correspond to outputs from functions.**

# 4. Overview of Functions

All of the functions called by the main script of the validation code are described as follows.

"**WT.obs.model.combination**," called in line 265:

- Inputs: "data.obs" and "data.model"
- Output: "obs.model"

This function combines observational and model data into the same data frame; the output consists of only observational and model data that have coinciding time stamps. In other words, if users have observational data from 2000 to 2010 in 1-minute intervals and model data from 2005 to 2010 in 5-minute intervals, the output is a data frame consisting of observational and model data from 2005 to 2010 in 5-minute intervals. If users have missing data for a certain time period, this data is not included because there is no common time stamp for model and observational data.

This function also provides quality control on the data, filtering out wind speed values that are above 40 m/s or below 0 m/s. Wind direction values below 0 degrees or above 360 degrees are discarded as well. Additionally, daily, hourly, monthly, and yearly columns are created in data.obs and data.model from previously created time stamps.

"**WT.timestamp**," called in line 297:

- Inputs: "long.obs.model" and "figure.dir"
- Output: "obs.model.timestamp"

This function creates a panel plot displaying time series of observed and modeled wind speeds. If users have wind direction and temperature data, time series of these are created as well. These plots help users visualize their data and easily indicate missing data. Also, time series confirm that users are working with the correct time intervals and magnitudes of parameters. Time series for wind speed are displayed in meters per second (m/s), for wind direction in degrees, and for temperature in the units specified in the configuration file. An example time series created from wind speed and wind direction data is shown in Figure 6.

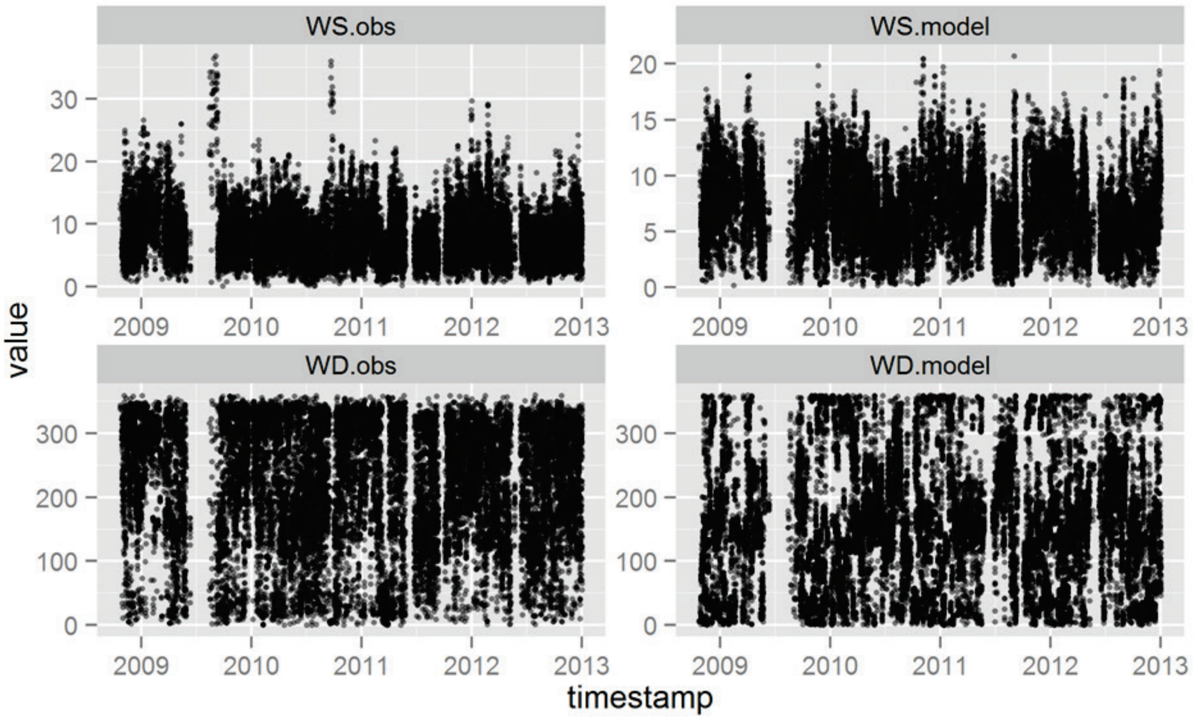


**Figure 6. Example time series of wind speed data in m/s ("WS.obs" and "WS.model") and wind direction data in degrees ("WD.obs" and "WD.model")**

"**WT.obs.model.Hourly**," called in line 309:

- Input: "obs.model"
- Output: "obs.model.Hourly"

This function uses the aggregate function to create average wind speed values for observational and model data for each day, hour, month, and year. Observational and model averages are merged into the same data frame to create a wind speed error column ("mean.WS.model" – "mean.WS.obs"). Input data is taken from "obs.model" and the output data frame is named "obs.model.Hourly." Users should not confuse this function with the similar approach used in "WT.error.metrics.Hourly.monthly." The output of "WT.obs.model.Hourly" is used to calculate a table of error metrics while "WT.error.metrics.Hourly.monthly" is implemented to create panel plots. An example output of "WT.obs.model.Hourly" for NWTC is displayed in Table 3.

**Table 3. Example Output of Function WT.obs.model.Hourly; Consists of Day, Hour, Month, Year, Mean Wind Speed for Observation and Models and Error Columns**

| Day | Hour | Month | Year | mean.WS.obs | mean.WS.model | Error |
|---|---|---|---|---|---|---|
| 1 | 0 | Apr | 2010 | 11.75 | 10.65 | -1.10 |
| 1 | 0 | Dec | 2009 | 12.51 | 8.91 | -3.55 |
| 1 | 0 | Feb | 2010 | 6.71 | 10.77 | 4.06 |
| 1 | 0 | Jan | 2010 | 7.77 | 7.84 | 0.07 |

"**WT.error.metrics.Hourly.monthly**," called in line 315:

- Input: "obs.model"
- Output: "error.metrics.Hourly.monthly"

WT.error.metrics.Hourly.monthly uses the aggregate function to create average wind speeds for observational and model data, using na.rm=TRUE, for each hour and month. In essence, it takes averages of wind speed for different time intervals based on the shortest interval measurements available. These aggregates are then merged by hour and month to get observational and model data in the same data frame. A bias column is then added that calculates wind speed error (model – observations) for each time interval. This column is named "bias" instead of "error" for panel plot labelling reasons. Root mean squared error (RMSE) of wind speed is then calculated with the aggregate function for each day, hour, month, and year. Additionally, a centered-root-mean-squared-error (CRMSE) column is calculated as CRMSE = $(\text{RMSE}^2 - \text{bias}^2)^{1/2}$. This data is used to create panel plots and is not used in calculating tables of error metrics. An example output data frame for the NWTC is shown in Table 4.

**Table 4. Example Data Frame from Output of Function WT.error.metrics.Hourly.monthly, Which Consists of Hour, Month, Mean Wind Speed for Model and Observations, Bias, RMSE, and CRMSE Columns**

| Hour | Month | mean.WS.model | mean.WS.obs | Bias | RMSE | CRMSE |
|---|---|---|---|---|---|---|
| 0 | Apr | 6.77 | 5.80 | 0.97 | 4.07 | 3.96 |
| 0 | Aug | 5.95 | 4.85 | 1.11 | 4.24 | 4.09 |
| 0 | Dec | 6.41 | 5.54 | 0.87 | 5.06 | 4.98 |
| 0 | Feb | 6.85 | 5.52 | 1.32 | 4.67 | 4.48 |

**WT.error.facets**, called in line 331:

- Input: "obs.model"
- Output: "error.facets"

Because the data frame “error.metrics.Hourly.monthly” contains information about bias, RMSE, and CRMSE, we convert “error.metrics.Hourly.monthly” to a long data format (“long.error.metrics.Hourly.monthly”) to create panel plots. “WT.error.facets” first creates and saves individual plots of diurnal bias, RMSE, and CRMSE averaged for each month of the year. The function also creates, saves, and returns a paneled plot of diurnal bias, RMSE, and CRMSE averaged for each month of the year. These panel plots allow the user to see trends in bias, RMSE, and CRMSE for individual months and over different hours of the day. An example for the NWTC is provided in Figure 7.

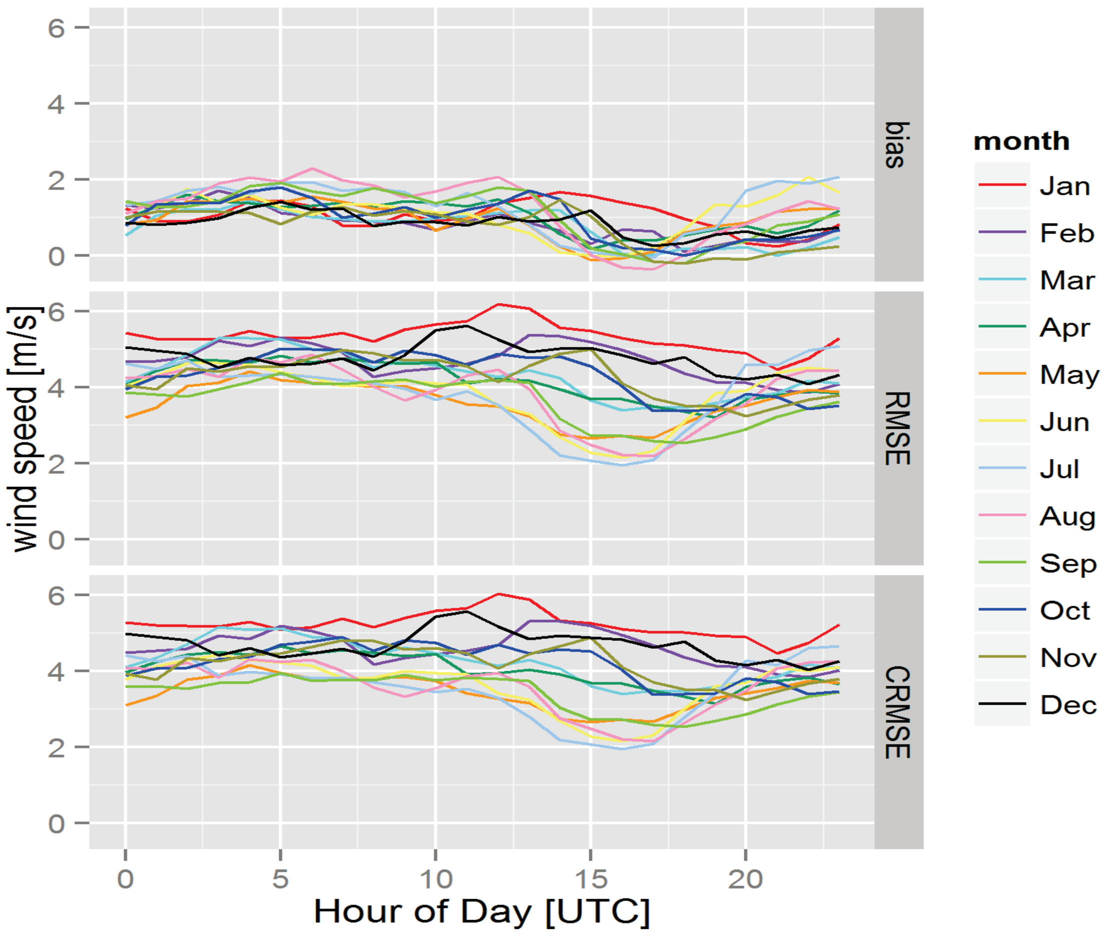


**Figure 7. Example output of function WT.error.facets; includes panels of diurnal bias (top), RMSE (middle), and CRMSE (bottom) averaged for each month of the year**

“**WT.month.Hour.facets**,” called in line 344:

- Inputs: “error.metrics.Hourly.monthly,” “long.error.metrics.Hourly.monthly”, and “figure.dir”
- Output: named (“error.month.Hour.facets”)

This function uses the long data format of long.error.metrics.Hourly.monthly to create panel plots of hourly and monthly averages of bias, RMSE, and CRMSE. Thus, instead of creating a diurnal bias plot with 12 lines (representing months); the new diurnal bias plot displays an average (one line) over different hours of the day. The saved file, "Hourly.facets.png," also contains averaged RMSE and CRMSE plots over each hour of the day. Similarly, "WT.month.Hour.facets" creates a panel plot of averaged bias, RMSE, and CRMSE over each month (this file is saved as "monthly.facets.png"). These panel plots allow the user to see overall trends in bias, RMSE, and CRMSE over hours and months. An hourly panel plot and a monthly panel plot for the NWTC are shown in Figure 8.

"**WT.windrose.facets**," called in line 365:

- Input: "temp.windrose.data"
- Output: "windrose.obs.model"

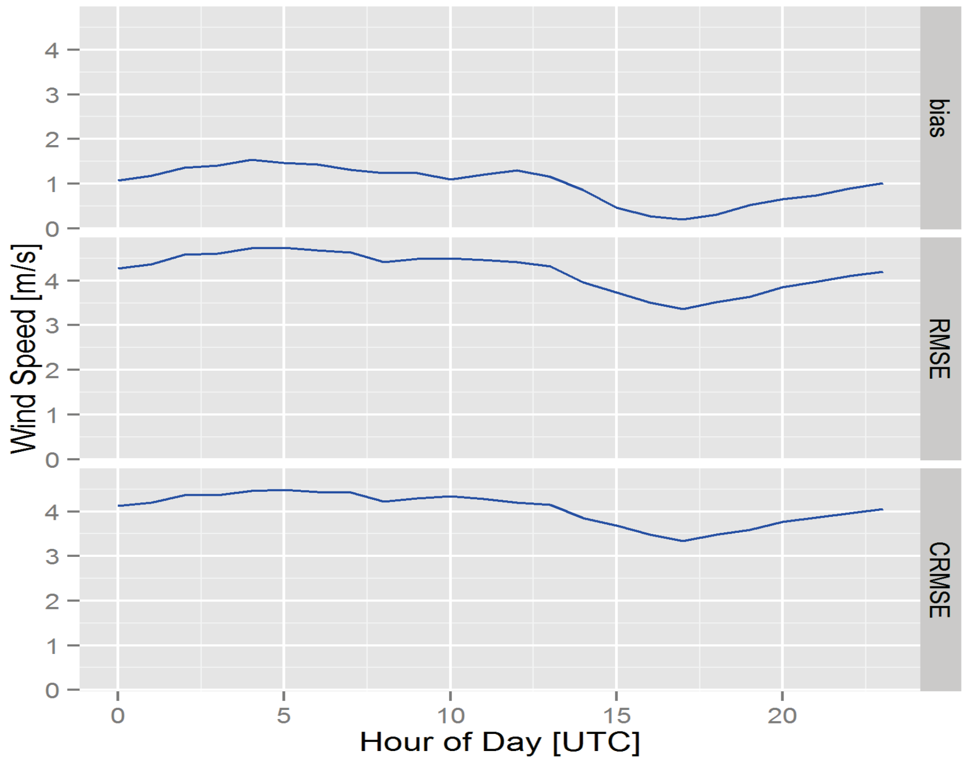
If the configuration file contains wind direction information, wind roses are created from observational and model wind speed and wind direction data. Before "WT.windrose.facets" runs, observed wind speed, observed wind direction, modeled wind speed, and modeled wind direction data are compiled into one data frame named temp.windrose.data. Running "WT.windrose.facets" first sends input data into another function called "WT.windrose." The rationale behind this approach is to eventually create a panel plot of observed and modeled wind roses. The function "WT.windrose" contains the code to construct wind roses, and the output from "WT.windrose" is sent back to "WT.windrose.facets." "WT.windrose.facets" then creates a panel plot of observed and modeled wind speeds, saving the figure as "Windrose.obs.model.facet.png."

Wind roses provide information about the magnitude of wind speed and wind direction for observational and model data (Figure 9). This provides the user an indication of the quality of their data, and if the directionality and magnitude of wind is what they expect. In the case of NWTC, the model and observation wind roses are almost identical, indicating that both the magnitude and direction of wind is accurately predicted by the WIND Toolkit data (Figure 9)

"**WT.windrose**," called within function "**WT.windrose.facets**":

- Input: "temp.windrose.data"
- Output: "p.windrose"

Produces separate plots of wind roses for observational and model data. The output is sent to "WT.windrose.facets" to create panel plots.



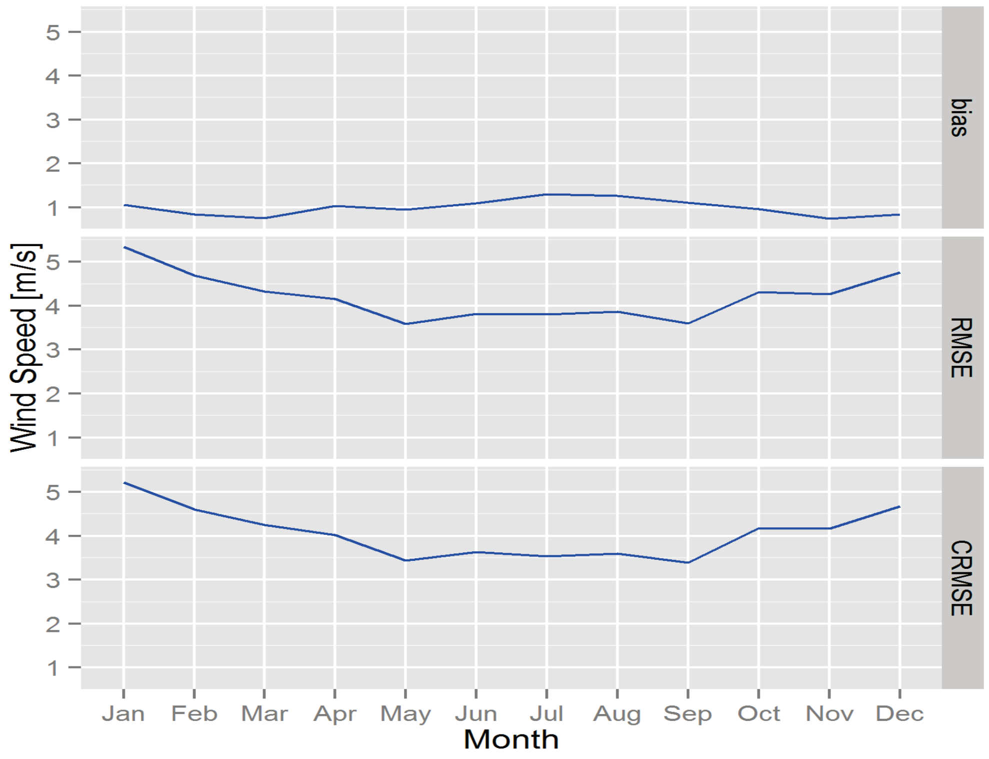


**Figure 8. Example output of function “WT.month.Hour.facets,” which returns panels of diurnal averaged bias, RMSE and CRMSE (top), as well as of monthly averaged biases RMSE and CRMSE (bottom)**

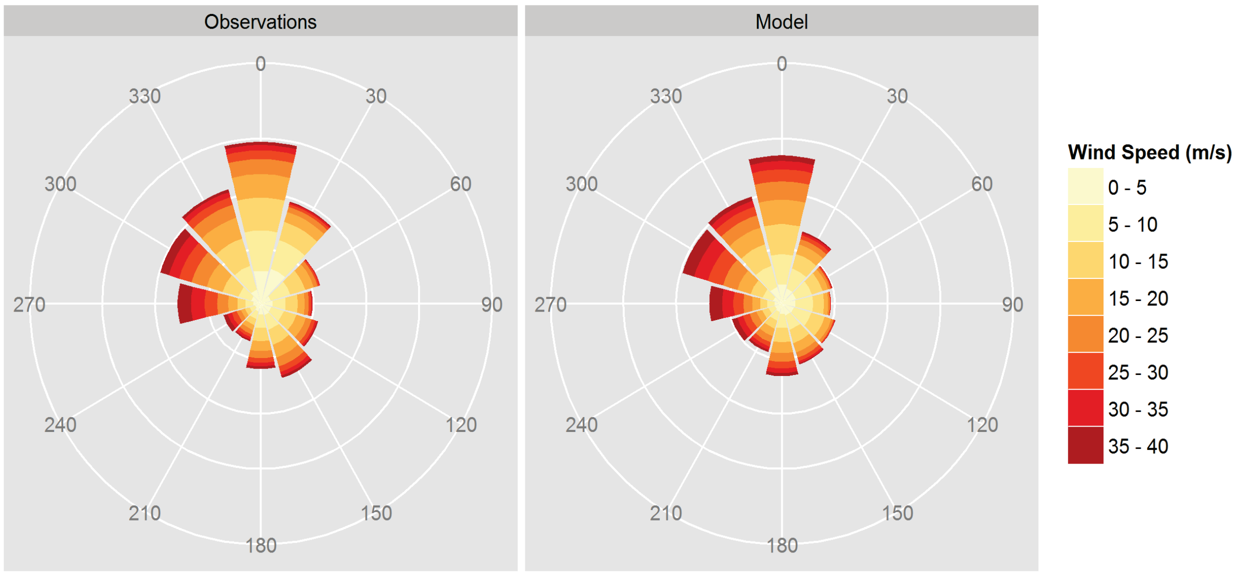


**Figure 9. Example output of function "WT.windrose.facets," which returns a wind rose of observational (left) and model data (right)**

"**WT.table**," called in line 386:

- Inputs: "data.interval" ("obs.model") and "data.time.avg" ("obs.model.Hourly")
- Output: table (data frame)

The function "WT.table" calculates error metrics from both interval averages and hourly averages. Interval averages are calculated from obs.model, which contains data in 5-minute intervals or the shortest available time interval. Hourly averages are calculated from means calculated in "obs.model.Hourly." "WT.table" calls functions defined in "WT.error.metrics.functions" to calculate bias, RMSE, mean absolute error (MAE) and percent error (pe). CRMSE is calculated as sqrt(RMSE$^2$ – bias$^2$). Both approaches to determine error metrics are included to compare values and show that the validation result only changes marginally. An example table for the NWTC is displayed below (Table 5).

**Table 5. Example Output of Function WT.table; Returns Error Metrics Based on Interval Averages (top) and Hourly Averages (bottom)**

| RMSE | Bias | CRMSE | MAE | pe |
|---|---|---|---|---|
| 4.28 | 1.00 | 4.16 | 3.01 | 110.45 |
| **RMSE** | **Bias** | **CRMSE** | **MAE** | **pe** |
| 3.89 | 0.99 | 3.76 | 2.71 | 78.60 |

"**WT.error.metrics.functions**," called within "**WT.table**" and "**WT.error.metrics.Hourly.monthly**":

- Inputs: obs, model, data, error (depends on the function)
- Defines bias, RMSE, MAE, and percent error (pe) functions

This function calculates bias, RMSE, MAE, and pe. Other functions are named accordingly; the function calculating bias is called "bias." If calculations require observed and modeled data, these data are combined into a data frame named "data." Missing data (NAs) are then omitted from this data frame to ensure that error metrics are calculated on the same amount of data.

The bias function requires inputs of observed and modeled data, calculates the mean of observed and modeled data separately, and returns the difference of the two means. The bias function is called within WT.table.

The RMSE function requires inputs of observed, modeled, and error data and calculates the RMSE. The default value for error is set to "NA." The RMSE function is called within "WT.table" and "WT.error.metrics.Hourly.monthly." If no error is specified, an error is calculated from observed and modeled data.

The MAE function also requires inputs of observed, modeled, and error data. MAE calculates an error term if no data is provided and returns the mean of the absolute error. MAE is called within "WT.table."

The pe function requires inputs of data, error, and observed wind speeds ("obs.WS"). The "data" input specifies the data frame to work with, as there are multiple data frames with error and observational wind speed ("obs.WS") columns. If the pe is calculated from interval data, a subset of the input data frame ("obs.model") is taken to include only nonzero wind speed values. The pe function is called within WT.table and calculates the pe in wind speed.

"**WT.histogram**," called in line 392:

- Inputs: "obs.model," "xlab," and "figure.dir"
- Ouput: Histogram

This function produces a histogram of wind speeds using observational and model data in "obs.model" (Figure 10). The histogram is constructed with the relative frequency, defined as the count of observations per total observations as a percentage, as the y axis. The x axis is labeled wind speeds (m/s), and the bars represent either observed (blue) or modeled data (red). Areas where observed and modeled data overlap are colored purple (Figure 10).

Histograms illustrate the distribution of observed and modeled wind speeds, as well as how modeled wind speeds compare to observed wind speeds. In the example for NWTC, the model over predicts observed wind speeds greater than 7 m/s and the model under predicts observed wind speeds lower than 7 m/s.

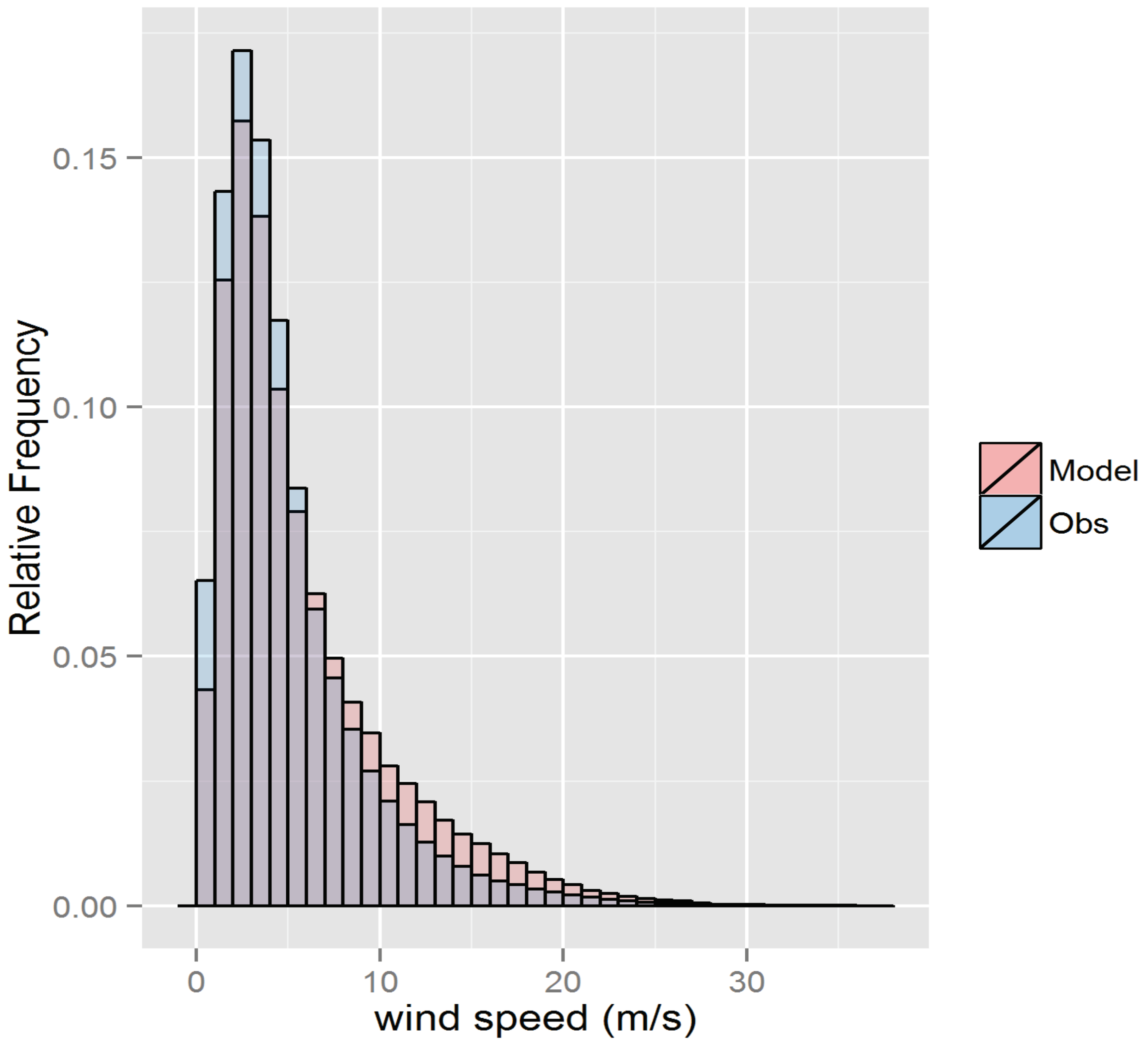


**Figure 10. Example output of function "WT.histogram," which returns a histogram of modeled (red) and observed (blue) wind speeds. Areas where observational and model data overlap are colored purple.**

"**WT.quantile.quantile**," called in line 399:

- Inputs: "obs.model," "figure.dir, xlab, ylab"
- Output: "p.qq"

This function creates a quantile-quantile plot of wind speeds using observational and model data in "obs.model." (Figure 11). Only model and observational data with coincident time stamps get mapped to the plot. "WT.quantile.quantile" first calculates quantiles from observed and modeled wind speed data. These quantile data were then plotted as a scatterplot with observed wind speeds on the x axis and modeled wind speed data on the y axis. A blue linear regression line was then fitted to the data and a black y = x line was added to the plot.

Quantile-quantile plots are useful because they inform users about the distribution and dispersion of observational and model wind speeds. If the quantile-quantile plot lies along a y = x line, the model and observational wind speeds are equal and come from the same distribution (Figure 11); however, if the majority of points lie above the y = x line, modeled wind speeds are more dispersed. Conversely, if the plot resides largely below the y=x line, observational wind speeds are more dispersed than modeled wind speeds. An example of the NWTC is included below (Figure 11).

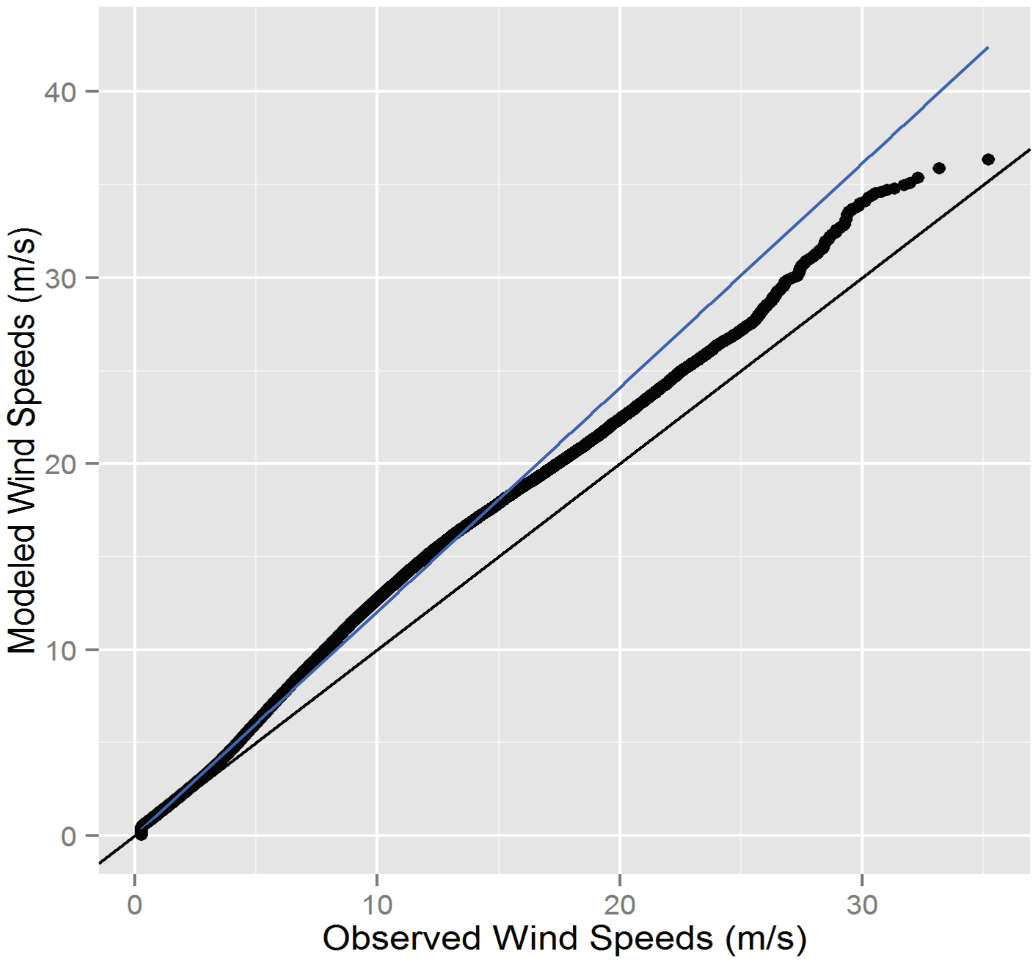


**Figure 11. Example output of function "WT.quantile.quantile," which returns a quantile-quantile plot of modeled and observed wind speeds. The black line is a standard y=x line, while the blue line is a linear regression line for the data. The data points are black circles.**

## References

1. Draxl, C.; Hodge, B.; Jones, W.; Getman, D.; McCaa, J. (2013). Overview and Meteorological Validation of the Wind Integration National Dataset. In review.
2. Nunalee, C.; Basu, S. (2013). Mesoscale modeling of coastal low-level jets: implications for offshore wind resource estimation. *Wind Energy*, **17, 8**, 1199–1216.
3. Potter, C.; Lew, D.; McCaa J.; Cheng, S.; Elchelberge, S.; Grimit, E. (2008). Creating the Dataset for the Western Wind and Solar Integration Study (U.S.A.). *Wind Engineering*, **32, 4,** 325–338.
4. Draxl, C.; Hahmann, A.; Pena, A.; Giebel, G. (2012). Evaluating winds and vertical wind shear from Weather Research and Forecasting model forecasts using seven planetary boundary layer schemes. *Wind Energy*, **17,** 39–55.
5. Carvalho, D.; Rocha, A.; Gomez-Gesteira, M.; Silva Santos, C. (2014). WRF wind simulation and wind energy production estimates forced by different reanalyses: Comparison with observed data for Portugal. *Applied Energy*, **117**, 116–126.
6. Skamarock, W.; Klemp, J.; Dudhia, J.; Gill, D.; Barker, D.; Duda, M.; Huang, X.; Wang, W.; Powers, J. (2008). A Description of the Advanced Research WRF Version 3. NCAR Technical Note NCAR/TN-475+STR, 10.5065/D68S4MVH.
7. Taylor, K. (2001). Summarizing multiple aspects of model performance in a single diagram. *Journal of Geophysical Research: Atmospheres*, **106**, 7183–7192

# Appendix

## How to Use Example Data to Perform a Validation

Before executing a validation, it is necessary to install R/R Studio. The central code to perform validations is called "main_script" and is located under the compressed zip folder "MainScript."

### *ExampleData Folder*

Example data is stored in a compressed zip folder titled "ExampleData." Example data is provided for the NWTC, Wasco, and Wentz. The configuration file, named "load.txt" is also provided. Inside any of the site-name folders, such as the "Wasco" folder, the necessary "data" and "figures" folders are present. Although these folders will originally be empty, results after completing Part 1 and Part 2 of the validation are provided. Inside the Wasco folder there are 71 files for observations and 1 file for model data. Observations for Wasco were taken in monthly intervals from 2007 to 2013, hence the 71 files. Although there are many observation files, they all need to be located under the site-name folder.

### *Defining the Location and Name of the "ExampleData" Folder in the Validation Code*

The compressed zip folder can be stored anywhere but the user must define the name and location of this folder in Line 91 of the code (see Section 3.3). Similarly, if the user changes the name of the configuration file ("load.txt"), this change needs to be recorded in line 92 of the validation code (Section 3.3).

### *Defining the Location and Name of the "Functions" in the Validation Code*

The user must also extract the functions used in the validation. These functions are stored within the compressed zip folder in a folder and named "functions." After extracting the "functions" folder, the location of the functions must be defined in Line 96 of the code (Section 3.5).

### *Performing Validation*

After defining names and locations of example data and functions, the validation code can be run. Completing the validation can be accomplished by pressing the "Run" button in R or selecting every line of code and pressing enter.

## Adapting Examples to Validate User Data

To perform additional validations, it is necessary to update the configuration file with appropriate information (Sections 3.1 and 3.2). Users are encouraged to be extremely careful entering data into the configuration file because this is a likely source of error. Users are encouraged to build off the example configuration file ("load.txt"). If users elect to create a new configuration file, the header of the new file must be the same as the example configuration file or the code will not work.

In addition to updating the configuration file, users must also create a new site-name folder that includes the name of each individual site the user wishes to validate (Section 3.4). In this folder, as shown in the examples, the user needs to create empty "data" and "figures" folders (Section 3.4). Inside the site-name folder, but not inside the "data" or "figures" folders, the user must place the observational and model data files to be used in validation. Users must also correctly define the location of functions so they can be sourced (Section 3.5).

Although model data for specific locations can be downloaded from the WIND Toolkit from NREL.gov (http://maps.nrel.gov/wind_prospector), the validation code can be used to calculate error metrics between any two datasets. In other words, it is not required for the WIND Toolkit to be used to perform validations.