The Optimal Locations for Solar Power Investment in Arizona

Alex Kuefler, Hector Lopez, Thao Nguyen
Department of Mathematics
Occidental College
Los Angeles, CA 90041 USA

May 5, 2015

Abstract

Solar installation sites must be chosen carefully in order to maximize return on investment for green energy projects. Given a database containing 27 potential sites in Arizona, we present a method for ranking them in order of their optimality. Three models of optimality, or score, are evaluated, where each is a function of solar panel efficiency, availability of solar resource and confidence in data for each site. Low uncertainty was obtained when comparing two of the models, which agreed on similar final rankings.

1. ***Introduction***

As interest in green alternatives to fossil fuel intensifies, many organizations have turned to solar radiation to meet their energy needs. However, the decision to build an installation for harvesting solar power is a potentially costly one. Although solar power is available anywhere, not all locations are made equal. The abundance of sunlight and effects temperature and humidity may have on photovoltaic cells make deciding the best place to put a solar panel tricky. Furthermore, the unpredictability of weather ensures that just because a site looks optimal today, it may not always be so. If the wrong decision is made, the consequences can be dire. Facilities that are unable to produce the projected amount of energy early in their lifespans run the risk of putting companies in debt to their financiers.[1]

As such, potential investors are conservative with their funding. Locations for solar panel placement are usually evaluated by the smallest possible solar energy return one can reasonably expect.[2] This figure, referred to as a P90 value, is the amount of a given resource that will be exceeded 90% of the time. P90 values are an important tool

[1] Vignola, F. et. al.
[2] Dobos, A. et. al.

for anyone wishing to assess a solar installation site, but they come with a built-in assumption that may not always be reasonable. That is, in order to compute the P90 value for a given location, one needs a set of *reliable* measurements of resources at that location. A large amount of historical data is needed to make accurate predictions about the long-term cycles of weather. Yet gathering many decades' worth of data introduces many decades' worth of measurement and transcription errors.

Our project combines physics, weather forecasting, and data science to determine both the abundance of solar resource as well as the degree to which measurements can be trusted for a potential site. Given a database of different locations, each characterized by years-worth of multivariate data, we present a method for ranking them in order of their optimality for gathering solar resource.

2. ***Methods:***

In order to rank each of the 27 stations in order of optimality, a score was assigned to each potential site. The score was the product of two quantities, referred to as *value* and *confidence*. Value is a function of meteorological data available at a given location and reflects how ideal that location appears, if the data are to be trusted. Confidence is a function of numerical features of the data itself, and reflect the degree to which the calculated value can be trusted. Because confidence is given as a real number between 0 and 1, it can be thought of as a weight that affects the final score.

It should also be noted that in the final version of the model, confidence c is not used to weight value, but used to weight the difference between the value V of a site and the mean value V of all sites. Using confidence to weight the difference of the mean has the effect of ranking high-confidence, low-value locations lower than low-confidence, low-value locations. This result is fortuitous, as confidence is intended to reflect the trustworthiness of the value. A higher degree of confidence should not raise our assessment of a station's optimality (as would be the effect of multiplying confidence by value itself), but should only adjust the assessment of the value. As such, sites whose value falls below the mean value will always have a negative score, whereas sites whose value exceeds the mean value will always have a positive score.

$$Score = c(V - \bar{V})$$

Once a score is found, sites are simply ranked in order of their scores, with higher scoring sites treated as being more optimal for solar power investment. It should be noted that each score, although represented as a real number, is only used for ordinal ranking. The actual quantity associated with the score is not considered relevant to the model. A precise explanation of the quantities value and confidence follows.

a. <u>Value V:</u>

Because we had a number of measurements available to us beyond solar radiation, our first step was to come up with a formula for *Value V* more sophisticated than the radiation P90 typically used. Since solar power depends heavily on radiation available at a given site, we still include the P90 value, but now we consider it alongside other factors. For instance, the efficiency of solar panels also depends on temperature. The manufacture efficiency of solar panels is mostly tested at a standard temperature of 25 degrees Celsius or 77 degrees Fahrenheit. The efficiency of solar panels can be written as the following function of temperature:

$$\eta = \eta_{T_{ref}} \left[1 - \frac{T_c - T_{ref}}{T_0 - T_{ref}} \right]$$ [3]

T_c is the current temperature (the variable).

T_{ref} is the temperature at which the solar cell efficiency is maximal, which is usually around 25C.

$\eta_{T_{ref}}$ is the efficiency of the solar panel at standard T_{ref} with solar radiation of 1000 watts per meters squared, and

T_0 is the high temperature, at which, the efficiency drops to 0. For crystalized silicon cells, it is about 270C. Since these are constant, the more the temperature increases from 25C, the lower the efficiency is, and vice versa. As such, a lower temperature implies greater efficiency. Since the $\eta_{T_{ref}}$ is the same for all locations, because we are using the same type of solar panels, we only consider the efficiency ratio $[\, 1 - \frac{T_c - T_{ref}}{T_0 - T_{ref}}]$ in order to evaluate the stations.

Furthermore, the solar radiation contains energy from many light spectra including both ultra violet and visible light. UV is the most energetic spectrum, and solar

[3] Skoplaki, E.

panels can only absorb energy from the visible light spectrum;[4] therefore, even if a cloudy place has a high value of solar radiation, the solar energy that can be used is very low. Hence, solar power plants receive the most radiation during clear sky weather. To predict cloudiness, we use relative humidity. Since relative humidity is defined as the ratio between actual vapor pressure and saturated vapor pressure, it reflects how saturated the water vapor in the air is. At 100% relative humidity, the air is saturated with water vapor, and there will be formation of water liquid. Clouds are a mixed form of water liquid and water vapor; therefore, the lower the relative humidity is, the less likely cloud formation is.

Considering the deleterious effect of cloud formation on the solar resource, we can finally come up with a *Value V* for each site. This figure is simply the product of the radiation with the efficiency ratio and the inverse of relative humidity.

$$V = Solar\ Radiation * \left[1 - \frac{Temperature\ (C) - 25}{245}\right] * \frac{1}{RelatiVe\ Humidity}$$

There are two ways to calculate V. The value of P90's, *V(P90)* is calculated from P90 of solar radiation, P90 of temperature, and P90 of relative humidity. The second way of calculating *V* is after calculating all of the *V* from daily solar radiation, temperature and relative humidity, we take the P90 value of *V* or *P90(V)* as the value.

b. <u>Confidence:</u>

Confidence measures the degree to which the value calculated for a given site can be trusted. It is an assessment based on numerical or historical features of the data itself, rather than the climatological conditions that the data are meant to portray. More formally, we defined confidence as the property of a system that allows one to make predictions about that system's future. High-confidence systems are deterministic, and their future can be easily predicted. Low-confidence systems are more stochastic or random, and confound attempts at long-term prediction.

Although three different metrics for confidence were proposed and assessed, each is a function of one or more of four variables we defined known as *errors*, *repeats*, *years* and *entropy*. Errors are defined as any time one of the meteorological variables we measured (i.e., solar radiation, temperature or relative humidity) exceeded the maximum

[4] Appendix A

or fell below the minimum quantity observed for the state of Arizona and were often represented in the dataset as a 999. Repeats can be thought of as another form of error, and represented an entire row of daily data being repeated. Although the quantities reported in a repeat row are, almost by definition, reasonable, these were assumed to be erred rows only when at least 26 of the 28 recorded variables (the columns representing year and day number were not considered) appeared multiple times together in the dataset. Errors and repeats are assumed to reflect some fault in the measurement equipment or data transcription at a given station, and should lower the confidence in a given station's data in any reasonable model.

The remaining variables, Years and Entropy, are more ambiguous. It seemed reasonable that these variables would inform our understanding of a site's confidence, but from the outside it was unclear if they would have positive or negative valences. Years is a percentage of the total years a stations has been on-line (where the oldest station was established in 1987)[5]. Because approximately 30 years of data is needed to accurately capture the long-term weather cycles of a given location[6], it initially appeared likely that a station being active for many years would have higher confidence. However, it could also be argued that older stations may use older equipment, which may be defunct or obsolete, and diminish the overall confidence in that location's data.

Information entropy, or Entropy, is the final variable of confidence. Entropy is a concept borrowed from information theory, which calculates the average amount of information in a dataset. Classically, entropy is defined for discrete random variables, such as the appearance of one the 26 letters of the Latin alphabet in a message. However, solar radiation, relative humidity, and temperature are continuous random variables. Instead, differential entropy is used to calculate the mean information of our meteorological variables. For a continuous-valued random variable X (such as solar radiance), with a probability density function $p(x)$, differential entropy is defined as

$$h(X) = -\int_{-\infty}^{\infty} p(x)\ln\big(p(x)\big)\,dx$$

[5] The University of Arizona
[6] Vignola, F. et al.

in natural units of information, or nats.[7] It was also unclear a priori how entropy would affect confidence. One might think that a greater amount of information in a data set would allow a modeler to make more accurate predictions about the system those data characterize. However, the mathematical definition of entropy is closer to uncertainty or surprisal, which would feasibly diminish our ability to make predictions. In other words, high-entropy systems tend not to behave monotonously, and systems with monotonous behavior are, if nothing else, predictable. One our four variables of confidence were chosen, the challenge of our many candidate models was to determine how each one should be used to influence our overall measure of confidence.

It should be noted that it might not be necessary to compute a confidence for many datasets, assuming one is reasonably trusting of the source of those data. However, the measurements reported on the Arizona Meteorological Network[8] are heterogeneous and noisy. Although the proposed model has been crafted specifically for this dataset, the four chosen variables of confidence are general enough to be applicable to other problem spaces. Due to the noisy nature of meteorological data, which is gathered over many decades by many instruments, we believe that confidence will be an important construct for measuring solar investment optimality from many datasets.

c. Candidate models:

Once value was defined and variables for confidence were selected, a number of potential models were generated. For compactness, we detail three particular model structures, of the general form

$$Score = c(V - \bar{V}),$$

however it should be noted that because value V can be chosen in two ways,[9] *V(P90)* and *P90(V)* , there were six candidate models in total.

Model 1. Our initial model included only the first three confidence variables, Errors, Repeats and Years. Here, the variables were thought of as a succession of weights to be applied to the value term, one after another. As in

$$c = (1 - e)(1 - r)(y)$$

[7] Michalowicz, J. V. et al.
[8] The University of Arizona
[9] Refer to the *Value V* section

where e is the percentage of days containing an error collected at a single station expressed as a decimal, r is the percentage of rows for all data at a station that have been repeated expressed as a decimal, and y is the percentage of years active, as a decimal. The variables e and r are subtracted from 1, because they were thought a priori to detract from our confidence in a site's data.

Model 2. In our second model, a site's confidence is a function of differential entropy.[10] Because entropy contains a summation over all data points in a data set (or in the case of a continuous random variable, an integral), it seemed redundant to include both Years and Entropy in the same model. If entropy sums over all data provided, we reasoned any information captured by the Years term would be contained implicitly in entropy. Similarly, because entropy measures surprisal, errors or noise seemed like they may also be factors of entropy. In symbols, this model for confidence was simply

$$c = h(X)$$

where $h(X)$ is the entropy of continuous random variable X, here representing meteorological measurements.

Model 3. Although we had quantified many features of the data which we thought influenced confidence, the first two models failed to account for all these in an impartial way. It was unclear how the four variables should be weighted or even if the presence of each one should increase or diminish the confidence. In order to allow more flexibility for how each variable factors into the final score, we then changed our model of confidence to a linear combination of our four variables such that

$$c = f(Ae + Br + Cy + Dh(X))$$

where A, B, C, and D are unknown parameters and f is a function mapping its input onto a real number between 0 and 1. f has the effect of forcing c to behave as a weight, as original intended.

The problem then reduced to fitting the parameters A, B, C, and D to ensure that c captures the confidence of the system as we had originally defined the term. Because we defined confidence as the property of a system that allows one to make predictions about

[10] Refer to the *Confidence* section

that system, we understood confidence as correlating with prediction accuracy. As such, finding the correct parameters was a process of producing a 27-entried vector $\vec{P}$ of prediction accuracies (where each entry gives the accuracy for a different site), and optimizing its correlation to a 27-entried vector $\vec{C}$. The entries of $\vec{C}$ are of the form

$$f(Ae_i + Br_i + Cy_i + Dh_i(X))$$

where e_i, r_i, y_i, and $h_i(X)$ are the empirical quantities from each site.

Formally, this model needed to optimize the objective function, Pearson's R Correlation[11],

$$R(P,C) = \frac{\sum_{i=1}^{n}(P-\overline{P})(C-\overline{C})}{\sqrt{\sum_{i=1}^{n}(P-\overline{P})^2 \sum_{i=1}^{n}(C-\overline{C})^2}}$$

by changing the parameters inside $\vec{C}$. This optimization was accomplished by gradient ascent[12], an iterative process that begins by initializing A, B, C, and D equal to 1 and computing the partial derivative of R with respect to each of the four parameters. Each parameter is incremented by the value of its partial derivative until convergence (i.e., a local maximum of the function R) is reached.

After optimization, the final form of the gradient ascent model was

$$c = f(-0.87e - 1.77r + 0.57y - 0.47h(X))$$

3. ***Results***

By graphing the models for each station, we developed a visual understanding about the behavior of and consistencies in our 6 models. Graph 1 shows that the degree of closeness between each value varies for each station. Presumably, the smaller the degree of closeness between values, the more consistent our models are. Therefore, for a given station, if the graph demonstrates a significant distance between each value, then the station is harder to predict since the values are varying, or unstable. We can generalize this idea: the consistencies in the models determine how confident we are about that station. If the values are significantly close, then the more confident we are about that

[11] Rice University et. al.
[12] Ng. A.

location. The inverse is also true. If the values are notably far apart, the less confident we are about it.

To determine the accuracy or precision of our six models, we found the uncertainty for each station. The equation that was used to find the uncertainty is given below.

$$Each\ site = \frac{Max(rank\ score) - Min(rank\ score)}{2} \quad (1)$$

Generally, uncertainty determines the error between values and allows us to judge the quality of our models. A small value translates to a small degree of closeness, and similarly, a large uncertainty indicates a large distance between values. Using this principle, we would use uncertainty to help measure confidence since we want the optimal locations to have small values for uncertainty. This value would suggest that our models were consistent.

We used mean uncertainty to observe whether a certain model significantly changes the behavior of our results. Mean uncertainty, which is the simplest statistical uncertainty to implement, can be calculated by summing all of the uncertainties and dividing by the number of stations. We use equation 1 to generate equation 2, mainly

$$Mean\ uncertainty = \frac{\Sigma\, Each\ site}{27} \quad (2)$$

According to Graph 1[13], model 2[14] with differential entropy, *V(P90)* Differential Entropy* and *P90(V) * Differential Entropy,* were slightly distant from the other 4 models. Thus, we calculated for mean uncertainty with and without differential entropy in order to highlight any notable differences between both values. In fact, the mean uncertainties with and without differential entropy were 2.83 and 2.13, respectively. These values illustrate that the uncertainty at each site is greater with models that included differential entropy, thus we removed model 2, and proceeded to create Graph 2[15]. Graph 2 helps us make more reasonable conclusions about the most optimal locations to place solar panels since it is easier to examine any consistencies and discrepancies in our models. These are fewer values to analyze, so it should be easier to determine what locations are the most reliable. Therefore, using this graph, we made a chart that lists the ideal locations to place

[13] See Appendix B
[14] Refer to the *Models* section
[15] See Appendix B

solar panels at the top. Notably, Mesa, Yuma North, and San Simon share low elevation levels, low precipitation, and high amount of solar radiation. Furthermore, their respective values were consistent with Graph 2, indicating that the confidence for these locations were reasonably high. [16]

4. ***Discussion:***

The distance between the ranks of one station reflects the consistency of the models. Since the graph 1 of all 6 ranks in appendix B has a large distance between the points for each station on each vertical line, we can conclude that some of the models are not consistent with each other. However, as we realized that the model 2 has the confidence solely as the differential entropy increases the uncertainties between the models the most. We can safely assume that these models are incorrect. Additionally, the confidence of these models has not been given a negative weight for differential entropy as the gradient ascent method indicated. We can conclude that model 2 can be safely discarded from our assessment. Models 1 and 3 show a greater degree of consistency. Due to the objectivity of the model 3, we use it as our final model of confidence.

Even though the final rankings are consistent with each other in their weather data and elevation location, we understand that there are limitations in our models that need to be improved. Our value function based on cloudiness prediction only loosely accounts for the relative humidity. Since cloudiness and weather prediction is more complicated, and involves more than relative humidity, an expansion of our research for a mathematical model of cloudiness and weather prediction would benefit the overall model. Furthermore, the profit of a solar power station can be critically affected by other logistical factors, such as maintenance fees. Therefore, further research could include these non-climatological factors into the value to significantly improve the model.

Furthermore, our confidence can be made more accurate with access to training data of existing solar power stations' energy production. We can use the training data for our error analysis by comparing it to our models' output. Additionally, we used ARIMA as an important feature in creating our confidence. Due to limited funding, we could only access free trials of software offering ARIMA features in limited time. Therefore, investing in more reliable ARIMA software would be helpful for improving our model.

[16] The final ranking for the 27 stations can be found in Appendix D.

<u>Appendix A: Choosing variables for Value V</u>

Solar radiation – P90 value

The exceedance probability P90 of solar radiation computes the value at which, 90% of anytime (in the past), the solar radiation was larger or equal to. However, since the time range of data collected for each station is varied, the exceedance probability of the younger stations would be less reliable than the older ones'. Therefore, we need to calculate the reliability of each station's P90 based on its age.

Furthermore, the solar radiation is not the absolute ranking value for a profitable station since the efficiency of solar panels also depends on other factors such as temperature and relative humidity.

Solar Panel Properties

1. How do Photovoltaics (PV) works?

Solar panels use photovoltaics to directly convert sunlight into energy. This process is an application of the photoelectric effect. Some materials, especially semiconductors have the property that allows them to absorb light photon and emit electric current. In an atom of pure silicon, electrons is attracted to the nucleus and has an energy bond between them. When external energy, such as light photon, is added onto the atom, a few electrons can break free using this excess energy and floating around, creating electric current.

The most popular photovoltaic cell that is used in solar panel is silicon cell, which takes up to about 94% of the market, due to its efficiency comparing to other metals; therefore, in this paper, we will assume that the facility uses silicon cell and based our model on this information.

2. Cloudiness and Relative humidity

Although the photovoltaic cells absorb sunlight to create electric current, not all wavelength of sunlight is used in the process. Silicon solar cell absorbs mostly light in the wavelength range from about 400 to 1100 nm[17] which is visible light.

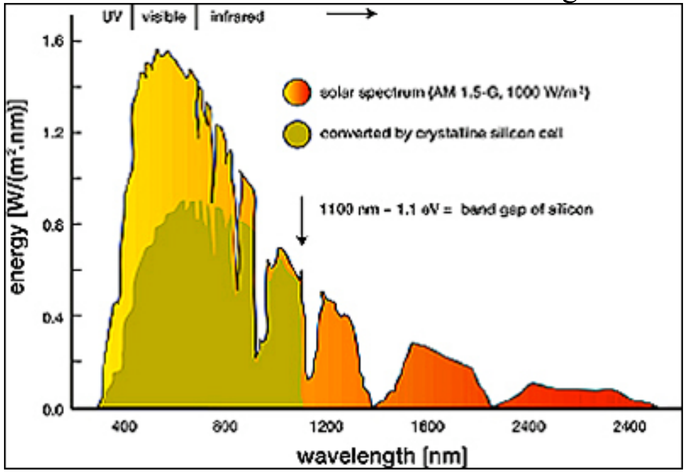


Figure 1. Silicon cell can only absorb a certain wavelength range

[17] Based on Figure 1 (source: Solar Cell Central)

This range does not include the most energetic light spectrum, which is Ultraviolet. Since solar radiation is composed of a large amount of energy from Ultraviolet spectrum, we need to make sure that the region that has a high amount of solar radiation also has clear sky.

Away to predict cloudiness or precipitation is based on relative humidity. Since cloud is the mix between water's vapor and liquid. Based on the phase shift of water[18], we can see that the dew point line between the gas phase and the liquid phase is where precipitation takes place, so the lower the combination of vapor pressure and temperature that stays in the gaseous or vapor phase, the less a significant chance of precipitation. An indication for this combination is relative humidity. Since relative humidity calculation is the quotient of real vapor pressure and the dew point pressure, the lower it is, the less chance of precipitation.

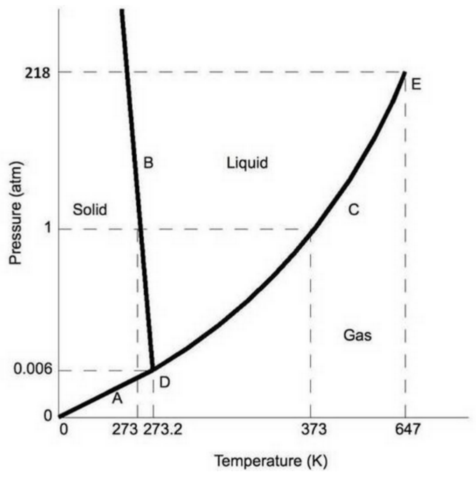


Figure 2. Water Phase Diagram based on Temperature and Vapor Pressure

3. Temperature and Efficiency

The efficiency

$$\eta = \frac{J_{max}V_{max}}{P_{light}}$$ [19]

of solar panel is determined by how much power it produces to the amount of light shines on it with

J_{max} is the current at the maximum power point,

V_{max} is the voltage at the maximum power point and

P_{light} is the power incident on the solar cell (the power from the light shining on it).

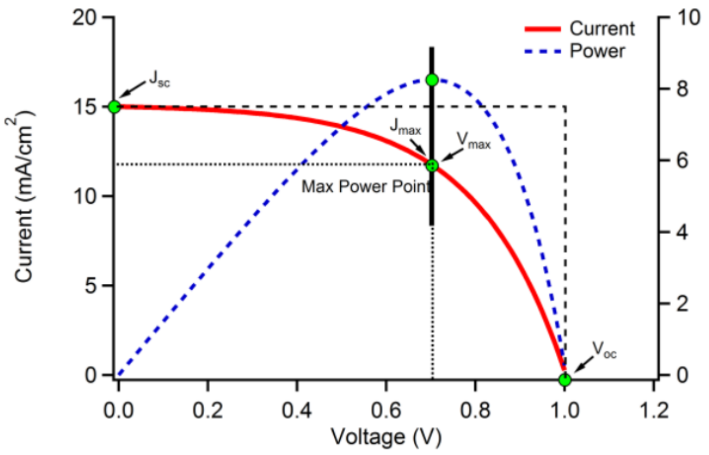


Figure 3. Current, Power vs Voltage in photovoltaic cell.[17]

[18] Based on Figure 2 [source: McKinnell, M., et. al.]

[19] Eisenmenger, N.

Based on figure 3, his equation can also be expressed as following

$$\eta = \frac{J_{sc}V_{oc}FF}{P_{light}} \text{ with}^{20}$$

J_{sc} is the current at short circuit (when V = 0),

V_{oc} is the voltage at open circuit (when J = 0) and

FF is the fill factor which describes how "square" the current-voltage curve is. It is the ratio between the two rectangles drawn in figure 3.

When the temperature is high, the solar cell is heated, so the short circuit current J_{sc} increases but both the open circuit voltage V_{oc} and fill factor *FF* decreases. Since the product of FF and V_{oc} decreases much faster than J_{sc}, the overall efficiency will decrease. The efficiency, therefore, can also obtained as a function of temperature T_c

$$\eta = \eta_{T_{ref}}[\,1 - \frac{T_c - T_{ref}}{T_0 - T_{ref}}] \text{ with }^{21}$$

T_c is the current temperature (the variable),

T_{ref} is the temperature, at which, the solar cell efficiency is maximal, usually at 25C,

$\eta_{T_{ref}}$ is the efficiency of the solar panel at standard T_{ref} with solar radiation of 1000 watts per meters squared, and

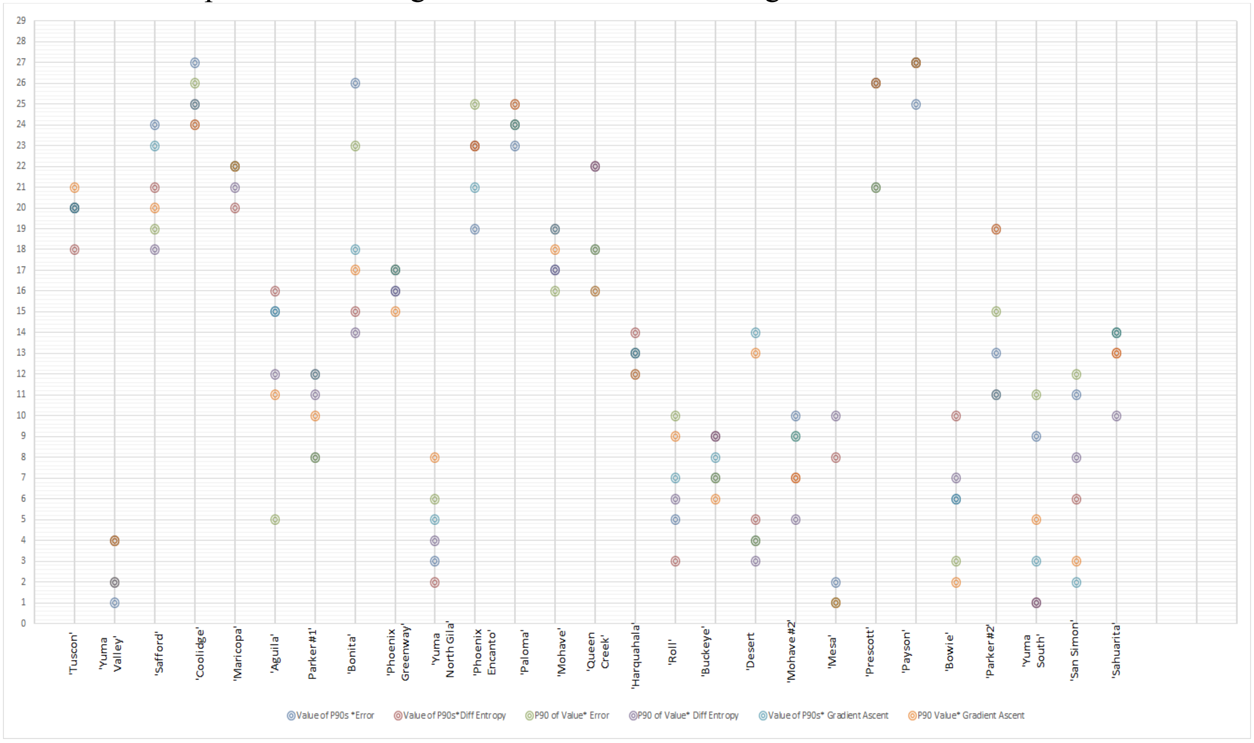
T_0 is the high temperature, at which, the efficiency drops to 0. For crystalized silicon cell, it is about 270C.

[20] Eisenmenger, N.

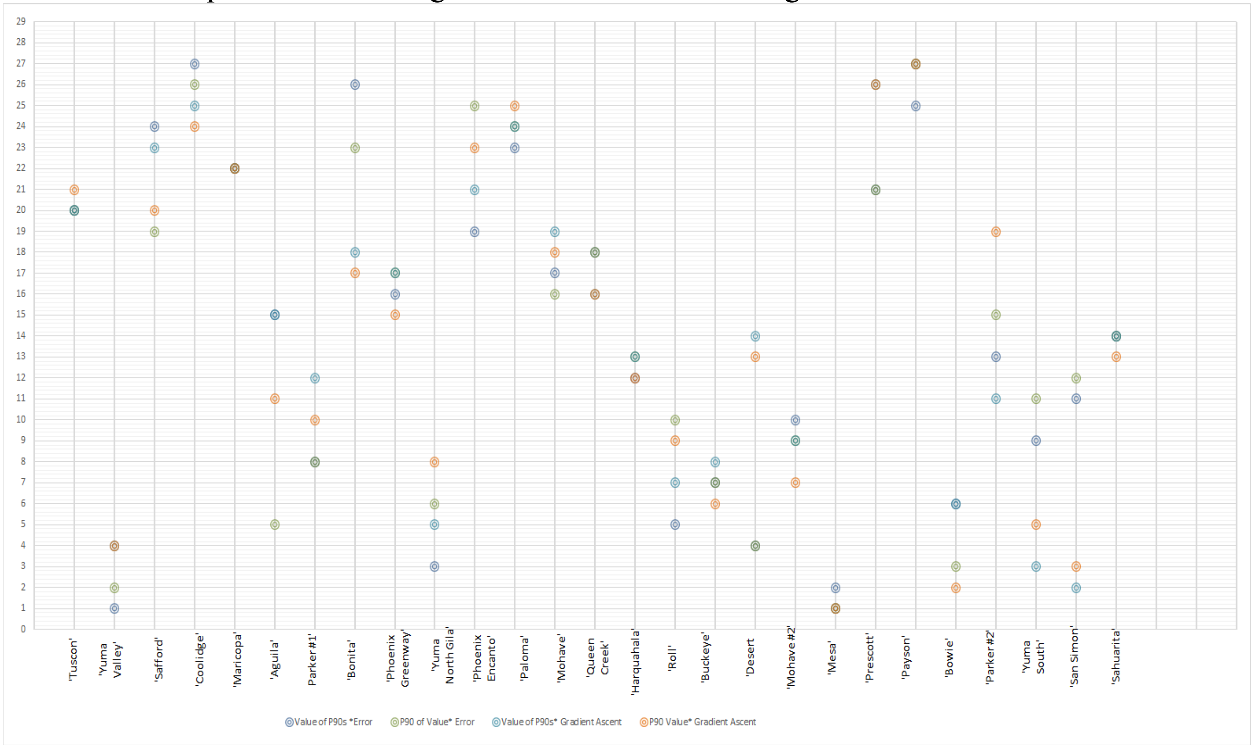
[21] Skoplaki, E.

Appendix B: Graph of different ranking models for 27 active stations

Graph 1: Six rankings of 27 active stations using all three models.



Graph 2: Four rankings of 27 active stations using model 1 and 3.



Appendix C: ARIMA model

ARIMA (Auto- Regressive Integrated Moving Average) uses three different models: autoregressive, moving average, and integration. Combining these three different components, ARIMA is able to analyze several aspects of the time series, and is able produce accurate forecasts through mathematical operations. Auto- regressive indicates that the output variable, Y_t, depends linearly on the previous values. "We forecast the variable of interest [Y_t] using a linear combination of *past values of the variable*."[20] We can mathematically translate it to equation 1 where $\phi 1, \phi 2, \ldots, \phi p$ are estimated coefficients.

$$Y_t = c + \phi 1 Y_{t-1} + \phi 2 Y_{t-2} + \cdots + \phi p Y_{t-p} + et, (1)$$ [22] .

The next model, integration, is used if the series is non- stationary. The reason for transforming a non-stationary series into a stationary one is explained by the fact that a stationary series maintains a constant mean, variance, and autocorrelation over time as opposed to the varying mean of a non-stationary series. Therefore, "a stationarized series is relatively easy to predict: you simply predict that its statistical properties [mean, variance, and autocorrelation] will be the same in the future as they have been in the past"(Duke University). Thus a constant mean is easy to forecast relative to an altering mean. Through a series of mathematical transformations, or differences, we can make our series stationary. Equation 2, which is represented as a difference between the current and previous value, is given below.

$$W(t) = Y_t - Y_{t-1} \ (2)$$ [23]

Finally, moving average accounts for forecast errors, or shocks. This error can be represented by the difference between the mean of the data set and the forecasted values. The mean is used since it remains constant for a stationary data set. In general, moving average accounts for previous mistakes, thus improving the accuracy of the future predictions. Equation 3 is composed of forecasts errors $e_t, e_{t-1}, e_{t-2}, \ldots e_{t-q}$, and parameters $\theta 1,\ \theta 2, \ldots, \theta q$.

$$Y_t = c + e_t + \theta 1 e_{t-1} + \theta 2 e_{t-2} + \cdots + \theta q e_{t-q} \ (2)$$ [24]

Process

In order to make predictions, we need to estimate 6 parameters: p, d, q, P, D, Q. These will generate an ARIMA model that will ultimately give us the forecasted values. The lower case parameters are non- seasonal while the upper case parameters are the seasonal components of our ARIMA model. We need to include seasonal elements because our data demonstrates a trend, mainly an oscillating trend. The chart below defines each parameter:

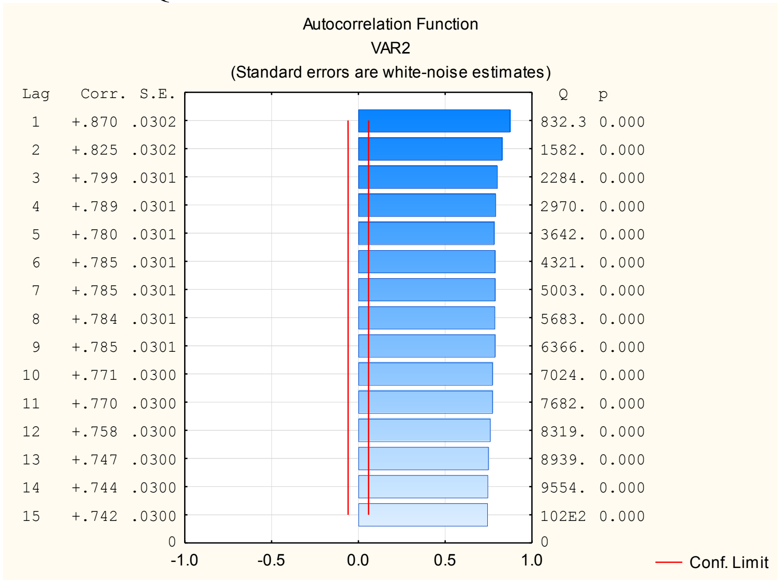
[22] Hyndman, J. et. al. [3]
[23] Hyndman, J. et. al. [4]
[24] Hyndman, J. et. al. [5]

p: number of non-seasonal autoregressive terms	P: number of seasonal autoregressive terms
d: number of non- seasonal differences	D: number of seasonal differences
q: number of non- seasonal moving average terms.	Q: number of seasonal moving average terms.

The estimations of these six parameters are found through a three stage process: identification, estimation, and forecasting. These stages require the application of several rules and practices that are listed in Duke University's *ARIMA models for time series forecasting*. Alongside the guidelines from this source, we will also use a statistical program, STATISTICA, which has a built in ARIMA feature. This program will provide you with the necessary elements, the autocorrelation function plot (ACF) and partial autocorrelation function plot (PACF), to come up with the most efficient ARIMA model. For each station, we used three years of daily data as our time series. We will look at how to generate an ARIMA model for a specific location, Queen Creek.

<u>Step 1: Identification</u>

This stage consists of classifying your series as stationary or non-stationary. First, we would need to inspect the autocorrelation plot (ACF) of the non- differenced series. The ACF for the series of Queens Creek is shown below.



```
Lag  Corr.  S.E.     Q     p
 1  +.870  .0302   832.3  0.000
 2  +.825  .0302   1582.  0.000
 3  +.799  .0301   2284.  0.000
 4  +.789  .0301   2970.  0.000
 5  +.780  .0301   3642.  0.000
 6  +.785  .0301   4321.  0.000
 7  +.785  .0301   5003.  0.000
 8  +.784  .0301   5683.  0.000
 9  +.785  .0301   6366.  0.000
10  +.771  .0300   7024.  0.000
11  +.770  .0300   7682.  0.000
12  +.758  .0300   8319.  0.000
13  +.747  .0300   8939.  0.000
14  +.744  .0300   9554.  0.000
15  +.742  .0300   102E2  0.000
```

"Rule 1: If the series has positive autocorrelations out to a high number of lags (say, 10 or more), then it probably needs a higher order of differencing" (Duke University)

The ACF demonstrates positive autocorrelation from lags 1 to 15, thus we would take a difference of the series. Now, we will use the differenced series to estimate the other parameters.

Step 2: Estimation

The second stage is estimating parameters, p, q, P, and Q. We have determined d, and because we want to avoid over-differencing the data, we set the value of D to 0. The remaining four parameters are determined by simultaneously looking at the autocorrelation function (ACF) and the partial autocorrelation function (PACF). These plots, however, contain 365 lags as opposed to 15. This can be explained by the fact that we want to observe and predict an entire year, or 365 lags. We have to consider two rules that would guide us toward the estimation of the two non- seasonal parameters, p and q.

If the autocorrelation function (ACF) of the differenced series displays a sharp cutoff and/or the lag-1 autocorrelation is negative ... then consider adding an MA term to the model."[23]

"If the partial autocorrelation function (PACF) of the differenced series displays a sharp cutoff and/or the lag-1 autocorrelation is positive… then consider adding one or more AR terms to the model."[25]

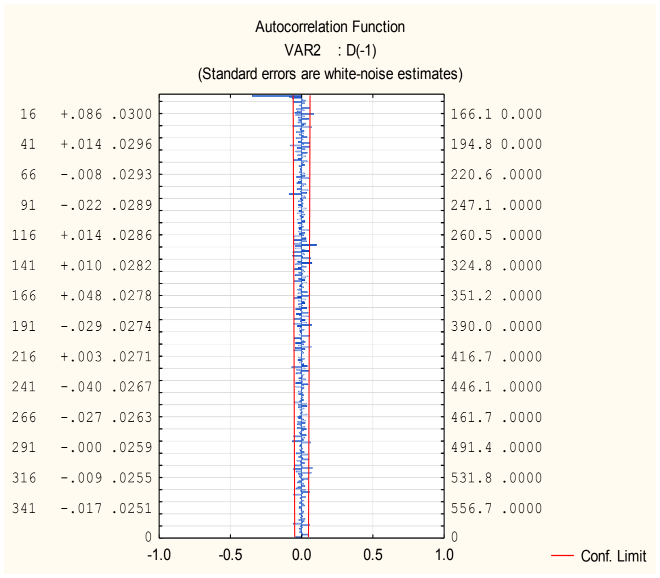


Figure 1: ACF (Autocorrelation Function Plot)

[25] Duke University

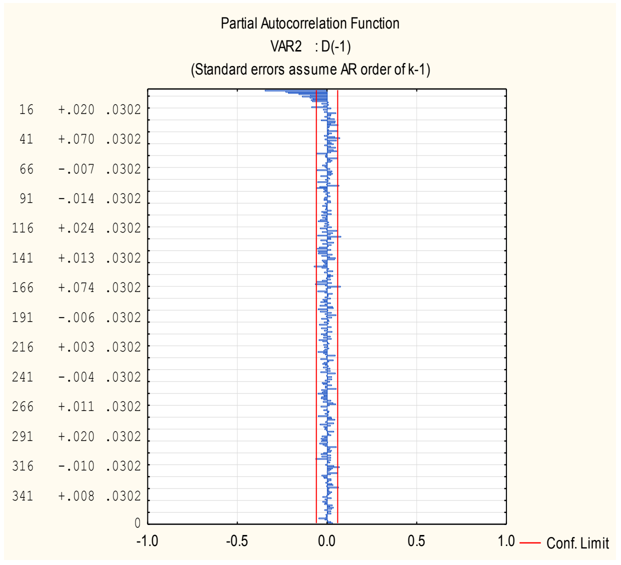


Figure 2: Partial Autocorrelation Function Plot (PACF)

Let us take a look at the autocorrelation function plot and the partial autocorrelation plot for one of the stations. Because the PACF displays a negative autocorrelation, then we do not consider adding an AR term. Also, we see that the ACF has a negative autocorrelation at lag 1, so we consider applying rule 7; however, we do not need to add an AR term if we can get a better estimation with no AR term. We can change the values of AR from 0 to 1 and determine which value gives a better estimation. Using the STATISTICA program, we are able to examine the standard error when AR is 1 or when AR is 0; therefore, we choose the value with the smallest standard error. Now, because we want to keep the seasonal pattern of the series, we find values for P and Q, which can be found by looking at the autocorrelation function.

"If the autocorrelation of the appropriately differenced series is positive at lag s, where s is the number of periods in a season, then consider adding an SAR term to the model. If the autocorrelation of the differenced series is negative at lag s, consider adding an SMA term to the model."[26]

Because the autocorrelation has 365 lags, it is difficult to implement this rule. This rule will be more efficient for an autocorrelation that contains fewer than twenty lags. But, generally, the values for P and Q are under 3. By varying the numbers for the 4 parameters and with the help of the STATISTICA program, we developed an ARIMA model with these parameters: p=0, d=1, q=0, P = 2, D=0, Q=1.

Step 3: Forecasting

The last step is forecasting. Finding the parameters creates an ARIMA$(p, d, q)X(P, D, Q)$ model, which will produce a predicted list of 365 values, or

[26] Duke University

an entire year. Below are graphs that overlap the actual 2015 solar radiation values with the predicted solar radiation outputs for the Queens Creek station. Overall, we were able to find 80% accuracy between the actual and forecasted.

Figure 3: Overlap of predicted and actual data in 2015 (Solar Radiation vs days from the beginning of 2012)

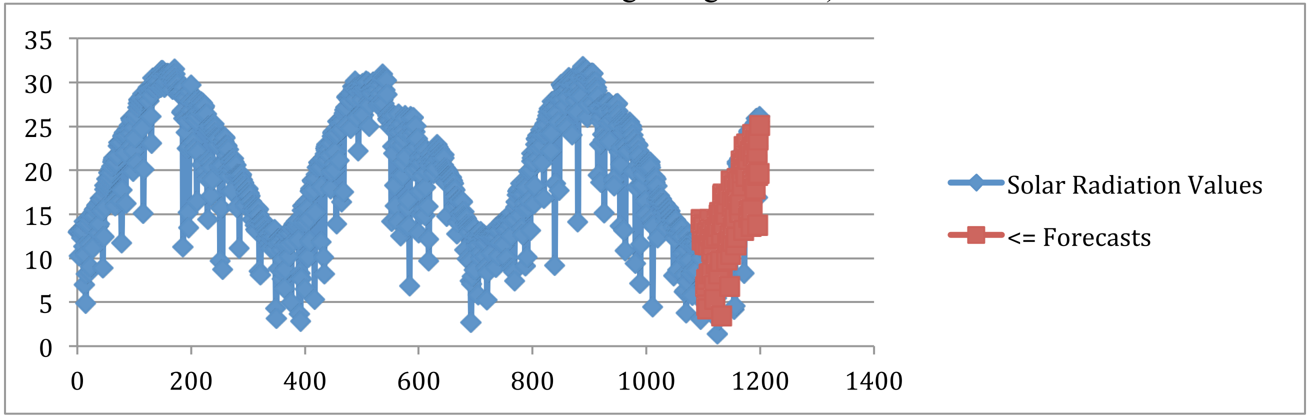
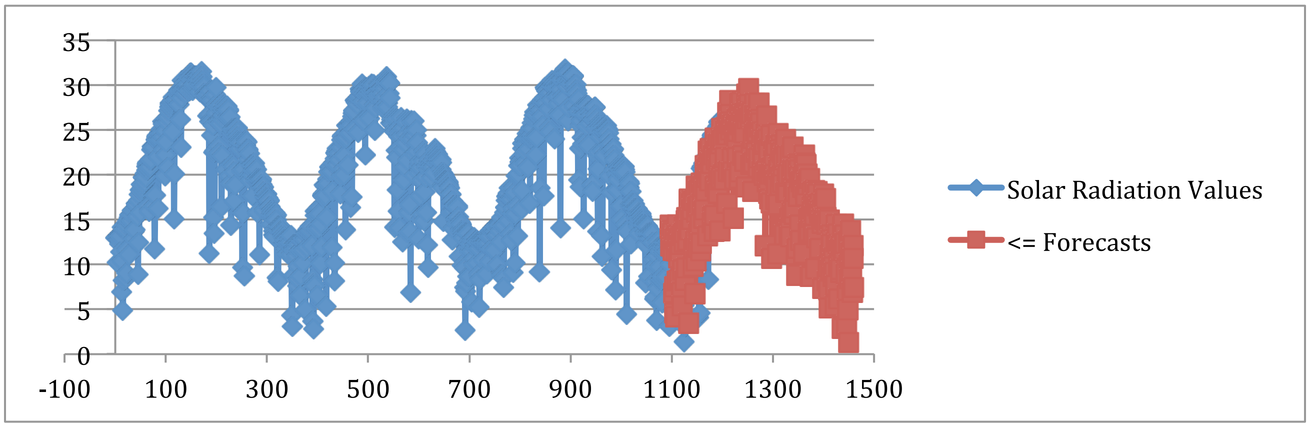


Figure 4: Prediction data in 2015 (Solar Radiation vs days from the beginning of 2012)



Appendix D: Table of Final Rankings

Final rankings produced for the AZMET dataset, based on confidence model 3, where Mesa is most optimal.

1. MESA	10. SAHUARITA	19. MOHAVE
2. SAN SIMON	11. PARKER #2	20. TUCSON
3. YUMA SOUTH	12. PARKER #1	21. PHOENIX ENCANTO
4. YUMA VALLEY	13. HARQUAHALA	22. MARICOPA
5. YUMA NORTH GILA	14. DESERT RIDGE	23. SAFFORD
6. BOWIE	15. AGUILA	24. PALOMA
7. ROLL	16. QUEEN CREEK	25. COOLIDGE
8. BUCKEYE	17. PHOENIX GREENWAY	26. PRESCOTT
9. MOHAVE #2	18. BONITA	27. PAYSON

Appendix E: Web Scraper in Python

```
import urllib
from urllib import urlopen
import sys
import re
import os
#Recovers daily data from the Arizona Meteorlogical Network website
(http://ag.arizona.edu/azmet/)
#and stores it locally in a directory hierarchy.

#Author: Alex Kuefler
#Date: February, 2015.

active_stations =
{"01","02","04","06","07","08","09","12","14","15","19","20","22","23","24","26","27",
"28",
              "29","05","31","32","33","35","36","37","38"}

text = 'http://cals.arizona.edu/azmet/data/0487rd.txt'
last_year = 2015

directory = 'dailies' #Title of the new directories to be created.

#Loop through every station provided in active stations.
for station in active_stations:
   directory = str(station)+'_dailies'

   if not os.path.exists(directory):
      os.makedirs(directory)

   os.chdir(directory)
```

```
    #Extract the the "Date on line" year for each station.
    page = urllib.urlopen('http://cals.arizona.edu/azmet/'+station+'.htm') #Access the data
page for current station.
    body = page.read()
    result = re.search('....\ *?\(Day', body)
    try:
        first_year = str(result.group())
    except:
        print(body)

    first_year = int(first_year[:4]) #Take only the last four digits (year)


    #Makes a .txt file for each year for the given station.
    for i in range(first_year,last_year):
        year = str(i)
        year = year[2:4] #We only want the last two digits. e.g., 1997 -> 97
        text = re.sub('(..rd)', year+'rd',text) #Hard coding rd (daily) for now.
        text = re.sub('/data/..','/data/'+station,text)

        u = urlopen(text)
        localFile = open(station+'_'+str(i)+'.txt', 'w')

        my_str = str(bytes.decode(u.read()))
        decoded_string = my_str.decode('string_escape') #Interpret expressions like \n in the
string.

        localFile.write(decoded_string)
        localFile.close()

    os.chdir("..") #Once inner loop terminates, go back to the parent directory
```

Appendix F: MATLAB Program for ranking

```
function [ stations ] = station_rank(  )
%Stores information about each station (e.g., score, rank, entropy,
P90's),
%and ranks the stations by our models.

%Must be placed in a directory containing a folder called 'data'. Must
run it after simple_scrape.py,
%but run simple_scrape INSIDE the folder 'data'.

%Authors: Alex Kuefler, Thao Nguyen
%Date: April 2015

    cd('data');
    station_files = dir;
    %Loop through every station
    for i = 4:length(station_files)
        cd(station_files(i).name);

        year_files = dir;

        all_years_SR = [];
        all_years_tempMax=[];
        all_years_tempMean=[];
        all_years_relhum=[];

        errors = 0;
        repeats = 0;

        %Loop through every year
        for j = 4:length(year_files)
            this_year_data = importdata(year_files(j).name);

            str = regexp(year_files(j).name,'\_....','match');
            this_year_num = str2double(str{1}(2:end));

            %Data were reformatted in 2003. Control statement if that
            %becomes important, which it isn't now.
            if this_year_num <= 2002
                sr_index = 11-2; %Subtract 2, because we use uqs, which
is missing first 2 values.
                tempMax_index=4-2;
                tempMean_index=6-2;
                relhum_index=9-2;

            else
                sr_index = 11-2;
                tempMax_index=4-2;
                tempMean_index=6-2;
                relhum_index=9-2;
            end

            uqs = unique(this_year_data(:,3:end),'rows'); %throw out
repeated rows

            %Concatenate this year to all years for each meteorlogical
            %variable.
            all_years_SR = [all_years_SR; uqs(:,sr_index)];
            all_years_tempMax = [all_years_tempMax;
uqs(:,tempMax_index)];
            all_years_tempMean = [all_years_tempMean;
uqs(:,tempMean_index)];
            all_years_relhum = [all_years_relhum; uqs(:,relhum_index)];

            repeats = repeats + length(this_year_data(:,1))-
length(uqs(:,1)); %Repeats are all rows that aren't captured by
unique().
```

```
            %errors = errors + length(find(uqs(:,sr_index) > 50));
            errors = errors + length(find(uqs(:,sr_index) > 50))...
                            + length(find(uqs(:,tempMax_index) >
50))...
                            + length(find(uqs(:,relhum_index) >
100))...
                            + length(find(uqs(:,tempMean_index) > 50));
        end

        ValueRaw=[];
        all_years_temp = ((all_years_tempMax-
all_years_tempMean)/2)+all_years_tempMean;
        ValueRaw = all_years_SR.*(1./all_years_relhum).*(1-
((all_years_temp-25)/(270-25)));

        %Calculate P90s for each variable.
        fx = sort_assign(all_years_SR,'sr');
        tx = sort_assign(all_years_temp,'temp');
        rx = sort_assign(all_years_relhum,'relhum');
        vx = sort_assign(ValueRaw,'sr'); %'sr' input arbitrary.
        sn = regexp(year_files(3).name,'..','match');

        stations(i-3).name = sn{1};

        %Assign the name of the town based on the station ID number.
        if strcmp(stations(i-3).name, '01')
            stations(i-3).title = 'Tuscon';
        elseif strcmp(stations(i-3).name, '02')
            stations(i-3).title = 'Yuma Valley';
        elseif strcmp(stations(i-3).name, '03')
            stations(i-3).title = 'Yuma Mesa';
        elseif strcmp(stations(i-3).name, '04')
            stations(i-3).title = 'Safford';
        elseif strcmp(stations(i-3).name, '05')
            stations(i-3).title = 'Coolidge';
        elseif strcmp(stations(i-3).name, '06')
            stations(i-3).title = 'Maricopa';
        elseif strcmp(stations(i-3).name, '07')
            stations(i-3).title = 'Aguila';
        elseif strcmp(stations(i-3).name, '08')
            stations(i-3).title = 'Parker #1';
        elseif strcmp(stations(i-3).name, '09')
            stations(i-3).title = 'Bonita';
        elseif strcmp(stations(i-3).name, '10')
        elseif strcmp(stations(i-3).name, '11')
        elseif strcmp(stations(i-3).name, '12')
            stations(i-3).title = 'Phoenix Greenway';
        elseif strcmp(stations(i-3).name, '13')
        elseif strcmp(stations(i-3).name, '14')
            stations(i-3).title = 'Yuma North Gila';
        elseif strcmp(stations(i-3).name, '15')
            stations(i-3).title = 'Phoenix Encanto';
        elseif strcmp(stations(i-3).name, '16')
        elseif strcmp(stations(i-3).name, '17')
        elseif strcmp(stations(i-3).name, '18')
        elseif strcmp(stations(i-3).name, '19')
```

```
            stations(i-3).title = 'Paloma';
        elseif strcmp(stations(i-3).name, '20')
            stations(i-3).title = 'Mohave';
        elseif strcmp(stations(i-3).name, '21')
        elseif strcmp(stations(i-3).name, '22')
            stations(i-3).title = 'Queen Creek';
        elseif strcmp(stations(i-3).name, '23')
            stations(i-3).title = 'Harquahala';
        elseif strcmp(stations(i-3).name, '24')
            stations(i-3).title = 'Roll';
        elseif strcmp(stations(i-3).name, '26')
            stations(i-3).title = 'Buckeye';
        elseif strcmp(stations(i-3).name, '27')
            stations(i-3).title = 'Desert Ridge';
        elseif strcmp(stations(i-3).name, '28')
            stations(i-3).title = 'Mohave #2';
        elseif strcmp(stations(i-3).name, '29')
            stations(i-3).title = 'Mesa';
        elseif strcmp(stations(i-3).name, '31')
            stations(i-3).title = 'Prescott';
        elseif strcmp(stations(i-3).name, '32')
            stations(i-3).title = 'Payson';
        elseif strcmp(stations(i-3).name, '33')
            stations(i-3).title = 'Bowie';
        elseif strcmp(stations(i-3).name, '35')
            stations(i-3).title = 'Parker #2';
        elseif strcmp(stations(i-3).name, '36')
            stations(i-3).title = 'Yuma South';
        elseif strcmp(stations(i-3).name, '37')
            stations(i-3).title = 'San Simon';
        elseif strcmp(stations(i-3).name, '38')
            stations(i-3).title = 'Sahuarita';

        end

        %extract p90 for rad, temp, and relhum
        stations(i-3).P90SR = xc_prob(0.90,fx);
        stations(i-3).P90TP = xc_prob(0.90,tx);
        stations(i-3).P90RH = xc_prob(0.90,rx);
        stations(i-3).P90Value = xc_prob(0.90,vx);

        %calculate the value
        stations(i-3).Value= stations(i-3).P90SR*(1/stations(i-
3).P90RH)*(1-((stations(i-3).P90TP-25)/(270-25)));

        %Calculates and assigns confidence variables.
        stations(i-3).percentErrors = errors/((length(year_files)-
2)*365)*100;
        stations(i-3).percentRepeats = repeats/((length(year_files)-
2)*365)*100;
        stations(i-3).percentYears = ((length(year_files)-2)/28);

        pd = fitdist(ValueRaw,'normal'); %Find the normal distribution
of the values.
```

```
        stations(i-3).differential_entropy =
log(pd.sigma*sqrt(2*pi*exp(1))); %Use normal distribution to find
entropy.

        stations(i-3).score = zeros(5,1); %Create a vector for the
scores, to be filled later.

        cd('..'); %Back out of current directory directory

    end

    mean_value = mean([stations.Value]);
    mean_P90 = mean([stations(i-3).P90Value]);

    %Rescales entropy, so it can be used as a weight on value in model
2.
    diff_entropy_weights =
mapminmax2([stations.differential_entropy],0,1);

    %Use gradient ascent algorithm to solve coefficients for confidence
    %variables.
    [A, B, C, D] =
gradient_ascent([stations.percentErrors]',[stations.percentRepeats]',..
.
    [stations.percentYears]',[stations.differential_entropy]');

    disp('Error coeff: ')
    disp(A)
    disp('Repeat coeff: ')
    disp(B)
    disp('Years coeff: ')
    disp(C)
    disp('Entropy coeff: ')
    disp(D)

    weights = [];
    for i = 1:length(stations)
        weights = [weights; (A*stations(i).percentErrors+...
            B*stations(i).percentRepeats+...
            C*stations(i).percentYears+...
            D*stations(i).differential_entropy)];

    end

    confidences = mapminmax(weights',0,1)';

    %Score 2 is P90-mean, weighted by entropy
    for i = 1:length(stations)

        %MODEL 1: Errors, Years, Repeats

        stations(i).score(1) = (stations(i).Value - mean_value)*((1-
0.01*stations(i).percentErrors) *...
```

```
            (1-0.01*stations(i).percentRepeats) *
(0.01*stations(i).percentYears));

        stations(i).score(3) = (stations(i).P90Value - mean_P90)*((1-
0.01*stations(i).percentErrors) *...
            (1-0.01*stations(i).percentRepeats) *
(0.01*stations(i).percentYears));

        %MODEL 2: Entropy

        stations(i).score(2) = (stations(i).Value -
mean_value)*diff_entropy_weights(i);

        stations(i).score(4) = (stations(i).P90Value -
mean_P90)*diff_entropy_weights(i);

        %MODEL 3: Gradient Ascent

        stations(i).score(5) = (stations(i).Value - mean_value)*...
        confidences(i);

        stations(i).score(6) = (stations(i).P90Value - mean_P90)*...
        confidences(i);

    end

    stations = rank_stations(stations);

end


function stations = rank_stations(stations)
%Assigns an ordinal rank to each station by each scoring model.

    scores1 = [];
    scores2 = [];
    scores3 = [];
    scores4 = [];
    scores5 = [];
    scores6 = [];

    for i = 1:length(stations)
        scores1 = [scores1, stations(i).score(1)];
        scores2 = [scores2, stations(i).score(2)];
        scores3 = [scores3, stations(i).score(3)];
        scores4 = [scores4, stations(i).score(4)];
        scores5 = [scores5, stations(i).score(5)];
        scores6 = [scores6, stations(i).score(6)];
    end

    sorted_scores1 = sort(scores1,'descend');
    sorted_scores2 = sort(scores2,'descend');
    sorted_scores3 = sort(scores3,'descend');
    sorted_scores4 = sort(scores4,'descend');
    sorted_scores5 = sort(scores5,'descend');
    sorted_scores6 = sort(scores6,'descend');

    for i = 1:length(stations)
        stations(i).rank = [find(sorted_scores1 ==
stations(i).score(1));
                find(sorted_scores2 == stations(i).score(2));
                find(sorted_scores3 == stations(i).score(3));
                min(find(sorted_scores4 == stations(i).score(4)));
                min(find(sorted_scores5 == stations(i).score(5)));
                min(find(sorted_scores6 == stations(i).score(6)))];
    end


end

function value = xc_prob(percent, sorted_SRs)
%Calculates the exceedene probability for percent such that percent =
90
%find the P90.

    fx2 = abs(sorted_SRs(:,2) - percent); %Find the value closest to
percent
    value = sorted_SRs(find(fx2 == min(fx2)));

    value = value(1); %In case it returns multiple dimensions.

end

function [ rads_and_fracs ] = sort_assign( solar_rads_vec, type )
%Sorts variables and assigns amount of total probability, allowing the
%P90's to be computed.

%Remove obvious errors from the data.
if strcmp(type,'sr')
    solar_rads_vec = solar_rads_vec((solar_rads_vec) < 51);
    solar_rads_vec = solar_rads_vec((solar_rads_vec) >= 0);
elseif strcmp(type,'temp')
    solar_rads_vec = solar_rads_vec((solar_rads_vec) < 50);
elseif strcmp(type,'relhum')
    solar_rads_vec = solar_rads_vec((solar_rads_vec) <=100);
end

%SR and temp vs. humidity need to be sorted differently, because they
%affect the model differently.
N = length(solar_rads_vec);
if (strcmp(type,'sr')||strcmp(type,'temp'))
    solar_rads_vec = sort(solar_rads_vec,'descend');
elseif strcmp(type,'relhum')
    solar_rads_vec = sort(solar_rads_vec,'ascend');
end

rads_and_fracs = [];
```

```
%Assigns the amount of total probability to each measurement once
sorted.
for i = 1:N
    entry = [solar_rads_vec(i) , (1/N)*i];
    rads_and_fracs = [rads_and_fracs; entry];
end


end

function [ A_g, B_g, C_g, D_g ] = gradient_ascent( A_vec, B_vec, C_vec,
D_vec )
%Optimizes Pearson's R Correlation with respect to confidence
variables.

syms A B C D

%Store the prediction accuracy at each station, found through ARIMA.
data = [
7   0.79509;
9   0.757548;
33  0.767;
26  0.8173;
5   0.8029;
27  0.7831;
23  0.7944;
6   0.7928;
29  0.7981;
20  0.7912;
28  0.7899;
19  0.795;
8   0.8163;
35  0.8146;
32  0.78;
15  0.7971;
12  0.7762;
24  0.8197;
1   0.8004;
22  0.7779;
4   0.7964;
38  0.7955;
37  0.7986;
14  0.7589;
36  0.7832;
2   0.7933;
31  0.7867];

data = sortrows(data,1);

%Need to optimize correlation between prediction accuracy vector X and
%confidence vector Y at every site.
X = data(:,2);
Y = A_vec.*A + B_vec.*B + C_vec.*C + D_vec.*D;

mean_X = mean(X);
```

```
mean_Y = mean(Y);

F_num = sum((X-mean_X).*(Y-mean_Y)); %Correlation numerator.
F_denom = sqrt((sum((X-mean_X).^2))*(sum((Y-mean_Y).^2))); %Correlation
denominator.

F = F_num/F_denom;

A_g = 1;
B_g = 1;
C_g = 1;
D_g = 1;
LR = 2; %Learning rate.

%This problem converges after 1000 iterations. Determined empircally.
for i = 1:1000
    A_g = A_g +
LR*eval(subs(subs(subs(subs(diff(F,A),D,D_g),C,C_g),B,B_g),A,A_g));
    B_g = B_g +
LR*eval(subs(subs(subs(subs(diff(F,B),D,D_g),C,C_g),B,B_g),A,A_g));
    C_g = C_g +
LR*eval(subs(subs(subs(subs(diff(F,C),D,D_g),C,C_g),B,B_g),A,A_g));
    D_g = D_g +
LR*eval(subs(subs(subs(subs(diff(F,D),D,D_g),C,C_g),B,B_g),A,A_g));

end

end
```

References

[1] Dobos, A., P. Gilman, and M. Kasberg., 2012: P50/P90 analysis for solar energy systems using the system advisor model. Preprints, 2012 World Renewable Energy Forum, Denver, CO, NREL, 1-6.

[2] Duke University, cited 2015: Introduction to ARIMA: nonseasonal models. *ARIMA models for time series forecasting.*[Available online at http://people.duke.edu/~rnau/411arim.htm]

[3] Hyndman, J., Rob and Athanasopoulos, G., 2012: 8.3 Auto-Regression Model. *Forecasting: principles and practices* [Available online at https://www.otexts.org/fpp/8/3] .

[4] Hyndman J., Rob and Athanasopoulos, G., 2012: 8.4 Moving Average Model. *Forecasting: principles and practices* [Available online at https://www.otexts.org/fpp/8/4]

[5] Hyndman, J., Rob and Athanasopoulos, G., 2012: 8.1 Stationarity and Differencing. *Forecasting: principles and practices* [Available online at https://www.otexts.org/fpp/8/1]

[6] McKinnell, M., Verhein, J., Yu, P., Chan, L.K., Dhaliwal, J., Bhela, S.,and Wong-Sing, D.,University of California, Davis, cited 2015: Phase Diagram. [Available online at http://chemwiki.ucdavis.edu/Physical_Chemistry/Physical_Properties_of_Matter/Phases_of_Matter/Phase_Transitions/Phase_Diagrams]

[7] Michalowicz, J. V., M. N., Nichols, and F. Bucholtz., 2008: Calculation of differential entropy for a mixed Gaussian distribution. Entropy., 10, 200-206.

[8] Ng, A., 2014: Supervised learning. CS229 Lecture Notes, 30 pp.

[9] Eisenmenger, N.,2011: The Temperature Dependence of Solar Cells. [Available online at http://www.scienceline.ucsb.edu/images/solarTempDepend]

[10] Rice University, University of Houston Clear Lake, and Tufts University, cited 2015: Computing Pearson's r. [Available at http://onlinestatbook.com/2/describing_bivariate_data/calculatation.html]

[11] Skoplaki, E., Palyvos, J.A., 2009: On the temperature dependence of photovoltaic module electrical performance: A review of efficiency/power correlations. Solar Energy., 83, 614-624. [Available online at http://dx.doi.org/10.1016/j.solener.2008.10.008]

[12] The University of Arizona, cited 2015: AZMET The Arizona meteorological network. [Available online at http://ag.arizona.edu/azmet/]

[13] Vignola, F., and C. Grover., N. Lemon and A. McMahan., 2012: Building a bankable solar radiation dataset. Sol. Energy., 8, 2218-2229.