User Manual
for
Uinta Basin Oil and Gas Emissions Model

Institute for Clean and Secure Energy

at the

University of Utah

December 15, 2016

Contents

I Model Overview **8**

1 Model Overview **9**

1.1 Introduction 9

1.2 Source Data Collection 10

1.3 Data Analysis 10

1.4 Monte-Carlo Simulation 11

1.5 Programming Language, Software, and File Structure 11

II `main` **13**

2 Global Options **14**

2.1 File Paths 14

2.2 Functions 15

2.2.1 `clipboard` 15

2.2.2 `inf_adj` 16

2.2.3 `CDFd` and `CDFq` 16

2.2.4 `asYear` 19

2.2.5 `NAoverwrite` 19

2.2.6 `calTstep` 19

2.3 Packages 20

2.4 Set and Load Options 21

3 Data Analysis **23**

3.1 `dogmDataUpdate` 23

3.2 `scheduleUpdate` 24

3.3 `EIApriceUpdate` 26

3.4 `leaseOpCostUpdate` 27

3.5 `drillingModelUpdate` 28

3.6 `ARIMAfitUpdate` 29

3.7 `EIAforecastUpdate` 33

3.8 `EIAerrorUpdate` 33

3.9 `EIAerrorUpdateFrac` 34

3.10 `DCAupdate` 36

3.11 `DCAupdateCDF` and `QfitDCAupdateCDF` 38

3.12 `reworkUpdate` 39

3.13 `DCAlnormUpdate` 39

3.14 `emissionUpdate` 44

4 Monte-Carlo Simulation **51**

4.1 Transition to MC Simulation 51

4.2 Energy Price Path Simulation 51

4.3 Well Drilling Schedule 53

4.4 Prior Wells 54

4.4.1 `priorProd` 54

4.4.2 `priorInfo` 54

4.5 Monte-Carlo Loop 54

4.5.1 Well Data Simulation 55

4.5.2 Production Simulation . . . 58
4.5.3 Lease Operating Costs . . . 58
4.5.4 Production Correction . . . 58
4.5.5 Activity-Based Emissions . . . 58
4.5.6 Equipment-based Emissions . . . 59
4.5.7 Totals . . . 65
4.6 Post-processing . . . 65
4.7 Save Results . . . 65

III `IO_options` 66

5 `IO_options` 67

5.1 Global Options . . . 67
5.2 Data Analysis Options . . . 70
5.2.1 `dogmDataUpdate` . . . 70
5.2.2 `scheduleUpdate` . . . 70
5.2.3 `EIApriceUpdate` . . . 71
5.2.4 `leaseOpCostUpdate` . . . 71
5.2.5 `drillingModelUpdate` . . . 72
5.2.6 `ARIMAfitUpdate` . . . 72
5.2.7 `EIAforecastUpdate` . . . 72
5.2.8 `EIAerrorUpdate` . . . 73
5.2.9 `DCAupdate` . . . 73
5.2.10 `DCAupdateCDF` . . . 75
5.2.11 `reworkUpdate` . . . 76
5.2.12 DCA Coefficient Distribution Fitting . . . 77
5.3 Monte-Carlo Simulation . . . 77

5.3.1 Energy Price Path Simulation . . . 77
5.3.2 Drilling Simulation . . . 79
5.3.3 Prior Production . . . 79
5.3.4 Well Data Simulation . . . 80
5.3.5 Production Simulation . . . 80
5.3.6 Activity-Based Emission Factors . . . 80
5.4 Post-Processing . . . 81
5.4.1 Plots . . . 81
5.4.2 Excel Export . . . 82

IV `EF_options` 83

6 `EF_options` 84

6.1 Global Options . . . 84
6.2 Equipment Type Options . . . 84

V `postProcess` 87

7 Post-Processing 88

7.1 Queries and Calculations in `postProcess` . . . 88
7.2 Excel Export . . . 88
7.3 Plots in `postProcess` . . . 89
7.4 Other Export Options . . . 90

VI Minor Scripts 91

8 Minor Scripts 92

8.1 R-Markdown Reports . . . 92

8.2 `README.md` . . . 93

VII Using the Model 94

9 Tutorials for Common Tasks 95

9.1 Model Installation and Setup . . . 95

9.2 Updating the Model with New Data . . . 97

9.3 Running the Model: Cross-Validation . . . 98

9.4 Running the Model: Prediction . . . 99

9.5 Exporting Results . . . 101

Bibliography 102

Appendices 104

A Nomenclature 105

A.1 User Manual Nomenclature . . . 105

A.2 Model Nomenclature . . . 109

List of Figures

1.1 Model Diagram. Acronyms used in figure stand for: oil and gas emissions inventory (OGEI), cumulative distribution function (CDF), cumulative probability table (CPT), and regression (Reg.). 10

2.1 Example `density` CDF output 17

2.2 `density` vs. `quantile` CDF comparison 18

3.1 Time-series decomposition of oil price history 30

3.2 Time-series decomposition of gas price history 31

3.3 Oil production from new wells using empirical CDF 40

3.4 CDFs for oil C_p decline curve coefficients by year 41

3.5 Linear regression of log-normal distribution parameters 42

3.6 Oil production from new wells using log-normal CDF 43

7.1 Example xy-quantile plot of oil price paths. 90

List of Tables

2.1 Packages utilized 21

3.1 Top 10 largest UDOGM fields 25

3.2 Required OGEI database tables 45

5.1 Mean and standard deviation of emission factors 81

A.1 User Manual Nomenclature 105

A.2 Model Nomenclature 109

Part I

Model Overview

Chapter 1

Model Overview

1.1 Introduction

The goal of this model is to forecast, with uncertainty estimates, all of the one-time and ongoing sources of emissions from oil and gas operations in Utah's Uinta Basin. The basic structure and design of the model is summarized in Figure 1.1. Broadly speaking, there are three stages to the modeling process: collection (and sometimes processing) of source data, data analysis, and finally performing a Monte-Carlo (MC) simulation. The major steps in each component of the model are discussed below, followed by a general overview of the model's file structure and code layout.

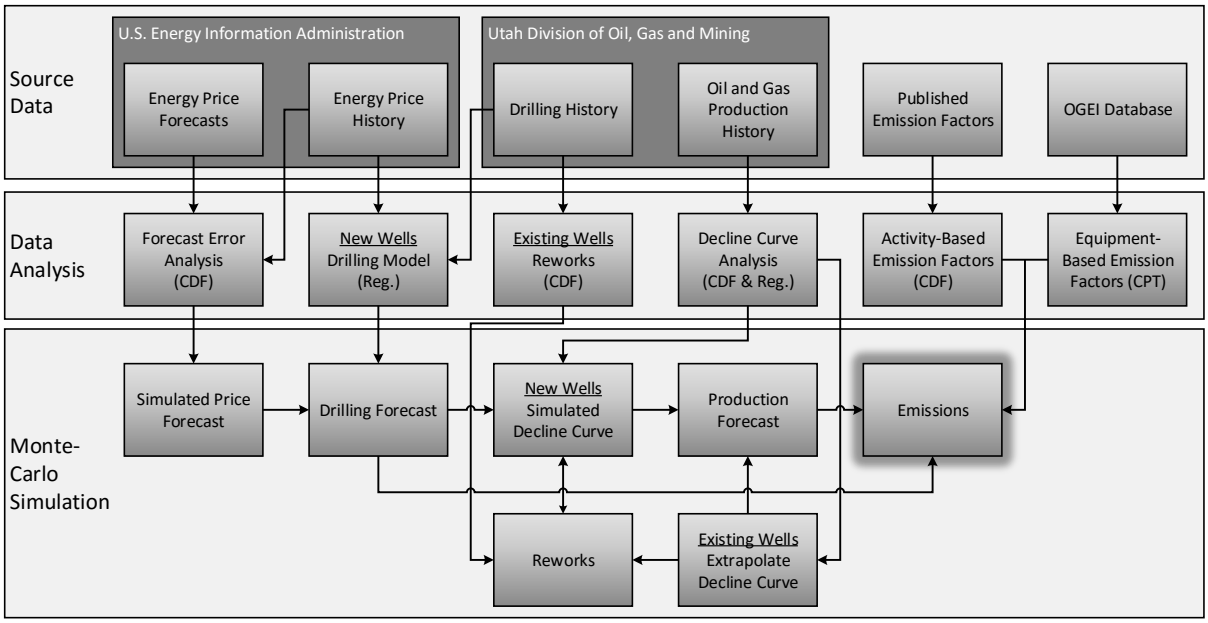

Figure 1.1: Model Diagram. Acronyms used in figure stand for: oil and gas emissions inventory (OGEI), cumulative distribution function (CDF), cumulative probability table (CPT), and regression (Reg.).

1.2 Source Data Collection

The two primary data sources used by the model are the U.S. Energy Information Administration (EIA) for energy price histories and forecasts, and the Utah Division of Oil, Gas and Mining (UDOGM) for well drilling and production histories. The sources of required data files are discussed in detail in the function descriptions of Chapter 3 wherever external data sources are used. EIA data sources are spread across a number of different websites and reports, consequently the user must pre-process and format the source data into comma-separated variable (csv) files. Template spreadsheets, built with Google Spreadsheets, are available and their use is discussed in the descriptions of the functions that use their data output. UDOGM database files require less pre-processing outside of R, however the user is still required to download the source database files from UDOGM's Data Research Center website (`https://oilgas.ogm.utah.gov/Data_Center/DataCenter.cfm`) and to run their executable files to extract the database files into a *.dbf file format. Source data for published emission factors (used for activity-based emission factor calculations) are taken from Wilkey et al. [2016]. Equipment-based emission factors rely on the oil and gas emissions inventory (OGEI) database, available from the Utah Division of Air Quality (UDAQ).

1.3 Data Analysis

Source data is primarily analyzed to generate:

- Cumulative distribution functions (CDF), which are a function that describes the probability that a certain term will have a given value between the minimum and maximum value for that term.
- Cumulative probability tables (CPT), which are a table containing (a) every unique combination of values from a larger database table and (b) a count of how many times each unique combination occurs.
- Least-squares regression, where a term is modeled as a function of a set of predictor variables with coefficients fitted to minimize the error between the model output and the term's observed values.

Most of the data analysis functions in the model are designed to produced CDFs for the terms used in the MC simulation. The primary functions that involve regression are the drilling schedule models, which fit a set of functions that predict the number of wells drilled as a function of energy prices, and the decline curve analysis functions, which fit both hyperbolic decline curves and cumulative production curves to the production histories for each well in the Uinta Basin. The data analysis functions are designed so that their results are saved to disk upon function completion; therefore they only need to be run whenever the source data files are updated.

1.4 Monte-Carlo Simulation

Monte-Carlo (MC) simulations are a form of modeling where terms in the model are picked randomly and independently from CDFs. If this random selection is repeated a sufficient number of times then the set of sample iterations will produce a range of outcomes that are representative of all the possible model outcomes (without having to actually solve for all possible combinations of the model input parameter terms). For each iteration (i.e. run) of the MC simulation, the model performs the following algorithm:

1. Generate an energy (oil and gas) price forecast
2. Calculate how many new wells are drilled in response to the energy price forecast
3. For every well (new and existing)
   (a) Pick well attributes (well depth, location, emission factors, etc.)
   (b) Calculate production rates
   (c) Calculate emissions from drilling and production
4. Sum together results for all wells to find total oil, gas, emissions, etc., for the given run of the MC simulation

Details of the MC simulation functions and their options are described in Chapter 4.

1.5 Programming Language, Software, and File Structure

The programming language used by the model is R, which is an open source language designed specifically for statistics and data analysis. R can be downloaded for your choice of operating system from `https://www.`

r-project.org/. While the only requirement for running the model is that the user must have R installed on their computer, the capabilities of the model are greatly enhanced with the following recommended software tools:

- RStudio (https://www.rstudio.com/)

  An integrated development environment (IDE) for programming in R. Greatly improves the user interface for coding in R in addition to providing version control tools for Git and SVN repositories.

- Git (https://git-scm.com/)

  For version control (i.e. tracking and managing code changes in model scripts and function files).

- Any cloud version control repository (Bitbucket, Github, etc.)

  Cloud hosting of the Git repository allows for code backup and for easy distribution of coding changes between multiple users.

- Any cloud data service (Dropbox, Google Drive, etc.)

  For storing source data files and data analysis results. Placing both the code repository and data files in the cloud is recommended (a) to improve the portability of the model, (b) to allow multiple users to work with the code simultaneously, and (c) to reduce the risk of code and/or data loss.

The model is comprised of four primary scripts. The first script is `main`, which performs the following functions (all of which are discussed in detail in Part II):

1. Establishes the global environment (e.g. setting file paths, loading libraries and functions, etc.).
2. Runs data analysis functions (or loads the results of a previous analysis).
3. Runs the MC simulation.
4. Saves specified results from the MC simulation.
5. Performs specified post-processing steps (primarily generating predefined plots).

The second script is `IO_options` , which contains and summarizes (nearly) all user input options and which is discussed in detail in Part III. Additional input options for equipment-based emission calculations are contained in the third script, `EF_options` (discussed in Part IV). The goal of splitting up the scripts this way is that the user should be able to set all of their desired modeling options and preferences in `IO_options` and `EF_options`, then run `main`, without having to search for or make any modifications to the function calls in `main` itself.

The fourth and final script is `postProcess`, which calculates the percentiles of the MC simulation results and generates a variety of predefined plots (e.g. production volumes of oil and gas, drilling rates, emissions by species and equipment, etc.). Detailed information about `postProcess` is available in Part V.

All other minor scripts not covered above are discussed in Part VI.

Part II

main

Chapter 2

Global Options

The global options are contained in Section 1.x of the `main` script. The primary purpose of this code segment is to:

1. Define the file path structure used in all subsequent sections of the model.
2. Load (or "source") all of the functions used by the model.
3. Load packages.
4. Set and load options.
5. If specified, load saved model results and stop further execution of the `main` script.

2.1 File Paths

The file structure for the model is divided into two sections: (1) a Git folder, which contains all scripts and functions that are tracked with version control, and (2) a "Dropbox" or cloud data service folder, which contains all pre- and post-processed data. The specific file paths and their intended purposes are listed below:

1. "raw"

   Location of all the raw data which is processed in the 2.x Data Analysis segment of `main`, most notably all of the UDOGM database *.dbf files.

2. "data"

   Location where all the processed data is saved after the 2.x Data Analysis segment.

3. "look"

   Only used in the `dogmDataUpdate` function, this folder contains a single csv file that specifies a name replacement pattern for use with formatting strings from the UDOGM database files.

4. "plot"

   Destination folder for all plots saved to portable document format (pdf) by the model, including well decline curve fits, MC results, etc.

5. "work"

   The working directory for the model (i.e. file folder that will be specified as the default location of any load or save file commands). By default this is set as the same directory as the location of the `main` and the other primary scripts in the model (`IO_options`, `EF_options`, and `postProcess`).

6. "fun"

   Location of all function files besides (other than those located under the "work" directory).

7. "plotfun"

   Sub-folder of "fun" folder, containing scripts called by `postProcess` that generate CDFs and boxplots of decline curve coefficients.

Assuming that future users keep the same file/folder structure used at the time of the model's release, the only items that need to be changed in the 1.1 Paths section of `main` are the locations of the Git (`pwd.git`) and Dropbox (`pwd.drop`) folders.

2.2 Functions

After setting the file paths, the next step in the global options segment is to source all of the functions that are used every time the model is run. The functions that are utilized on every run are (a) the functions in the MC simulation (all code sections labeled as 3.x), and (b) utility functions that are used repeatedly throughout the model. The functions used in Section 3.x are discussed in Chapter 4. The global utility functions are discussed below.

2.2.1 `clipboard`

This function copies an object in R (value, vector, data.frame, etc.) to the clipboard so that it can be pasted into a different program, and in particular, for copy/pasting data from R into Excel. The function is based on an article by Landgraf [2014], with modifications so that it works on a variety of operating systems (the default operating system is Windows, but other options enable use of the function in Mac and Linux). To use, simply place any R object inside the function call:

```
clipboard([any object in R])
```

Note that by default column names are also copied (suppress by including `clipboard(..., col.names = FALSE)` in the function call). Additionally, there are size limits that prevent copying large objects to the clipboard. If you encounter the size limit, an error message will be displayed and another exporting option, such as `write.csv`, must be used.

2.2.2 `inf_adj`

This function inflation adjusts a cost value using Equation 2.1:

$$C = C_o\left(\frac{I}{I_o}\right) \tag{2.1}$$

where C is the cost, I is the cost/inflation index, and the subscript o indicates the base value of C or I. The only cost index used in this model is the Consumer Price Index (CPI) [U.S. Bureau of Labor Statistics, 2015].

2.2.3 `CDFd` and `CDFq`

These two functions are designed to calculate the CDF for set of values stored in a vector object. `CDFd` accomplishes this task by:

1. Calculating the probability density function (PDF) calculated by R's `density` function. The additional inputs in `CDFd` are passed to the `density` function call.
2. Taking the cumulative sum of the PDF and multiplying it by the PDF's step size.
3. Dividing the cumulative sum by the maximum value in the summation to normalize the result.

Initially, `CDFd` was the sole method used for estimating CDFs. However there is a subtle problem with `CDFd`. The `density` function estimates the PDF by drawing a Gaussian distribution (or other user-specified distribution) over each point in the input vector, and then sums together the overlapping distributions to determine the total PDF. As a result the PDF will be nonzero outside of the range of values in the input vector. For example, suppose that the input vector is the initial production rates for oil (b_o in the hyperbolic decline curve equation or C_p in the cumulative production equation, which are discussed later in detail in Section 3.10), and further suppose that the values (generated from a random normal distribution with mean = 50 and standard deviation = 20) were as follows:

32 39 35 31 61 53 35 17 7 17 36 74 25 22 61 65 60 37 51 18

A plot of the CDF for this sample set is shown in Figure 2.1. Note that the CDF curve extends to negative initial production rate values (the cumulative probability of a negative number occurring in for this sample set is 2%), which is an impossible result (no well should have negative production volumes). The "from" and "to" inputs in `CDFd` were included specifically to fix this issue by clipping the results from `density` to just the range between "from" and "to." However it was later discovered that limiting the output range doesn't change the PDF result, and normalizing to the limited range skews the CDF so that random draws from that CDF won't reproduce the distribution of the input vector.

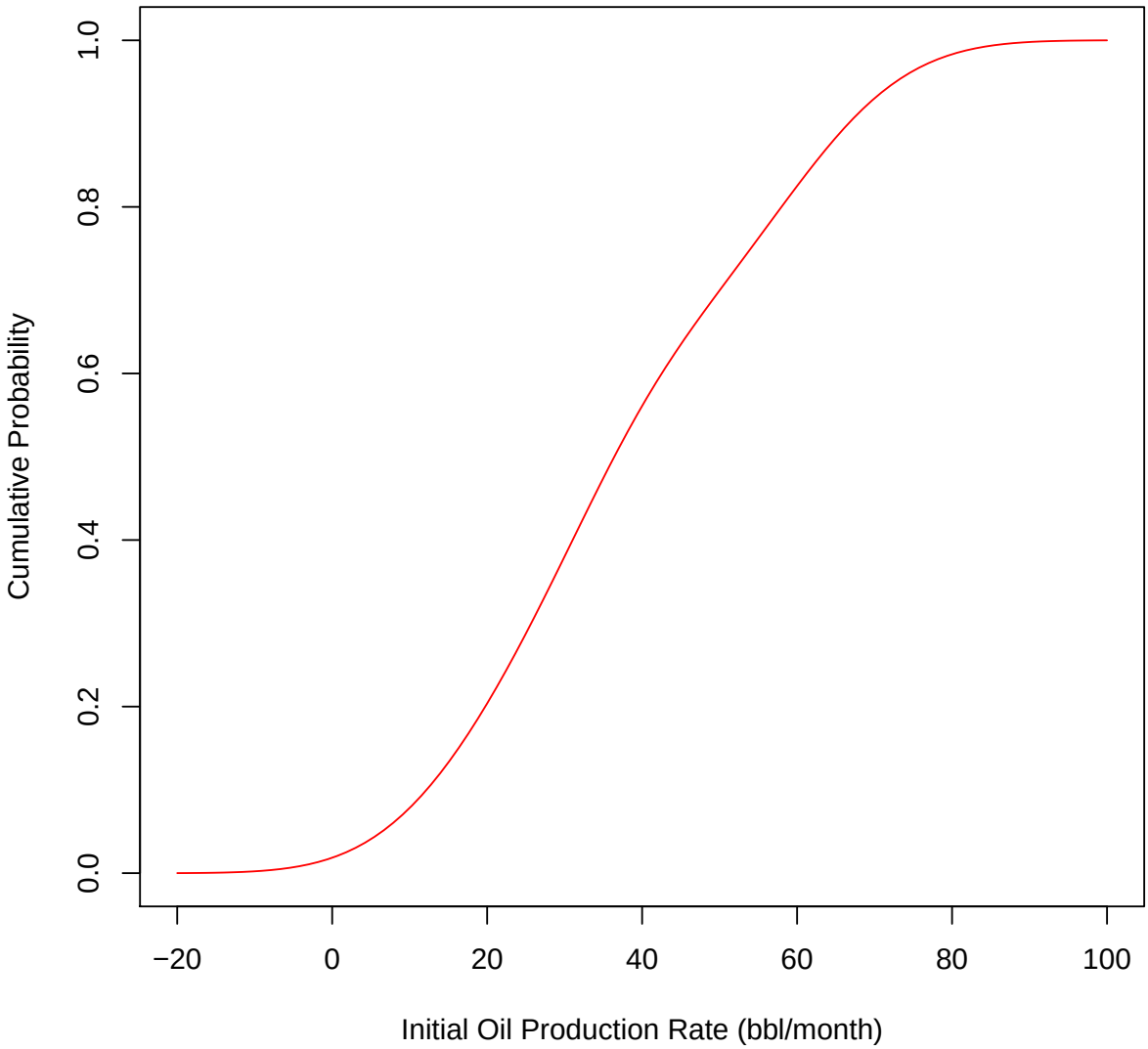

Figure 2.1: Example plot of the cumulative distribution function (CDF) calculated by R's `density` function.

After discovering this issue, an alternative method for generating CDFs using another function in R, `quantile`, was written. The `quantile` function returns the value of the input vector's CDF at a specified cumulative probability. For example, the value of the example initial oil production vector specified above at a cumulative probability of 0.5 (i.e. the median or 50th percentile) is 35.5. However any number of probability points can specified, so by simply inputing the full range of cumulative probabilities (from 0 to 1), `quantile` will return the CDF. The `CDFq` function does exactly that and reformats the result to match the output of the `CDFd` function. A comparison of the CDFs calculated by both `CDFd` and `CDFq` from the example input vector is shown in Figure 2.2. While the output from `CDFq` looks choppy (due in this instance

to the low number of data points in the input vector), it appropriately limits the CDF to the minimum and maximum values in the input vector (7 and 74), and any random draws from it will reproduce the same distribution as the input vector.

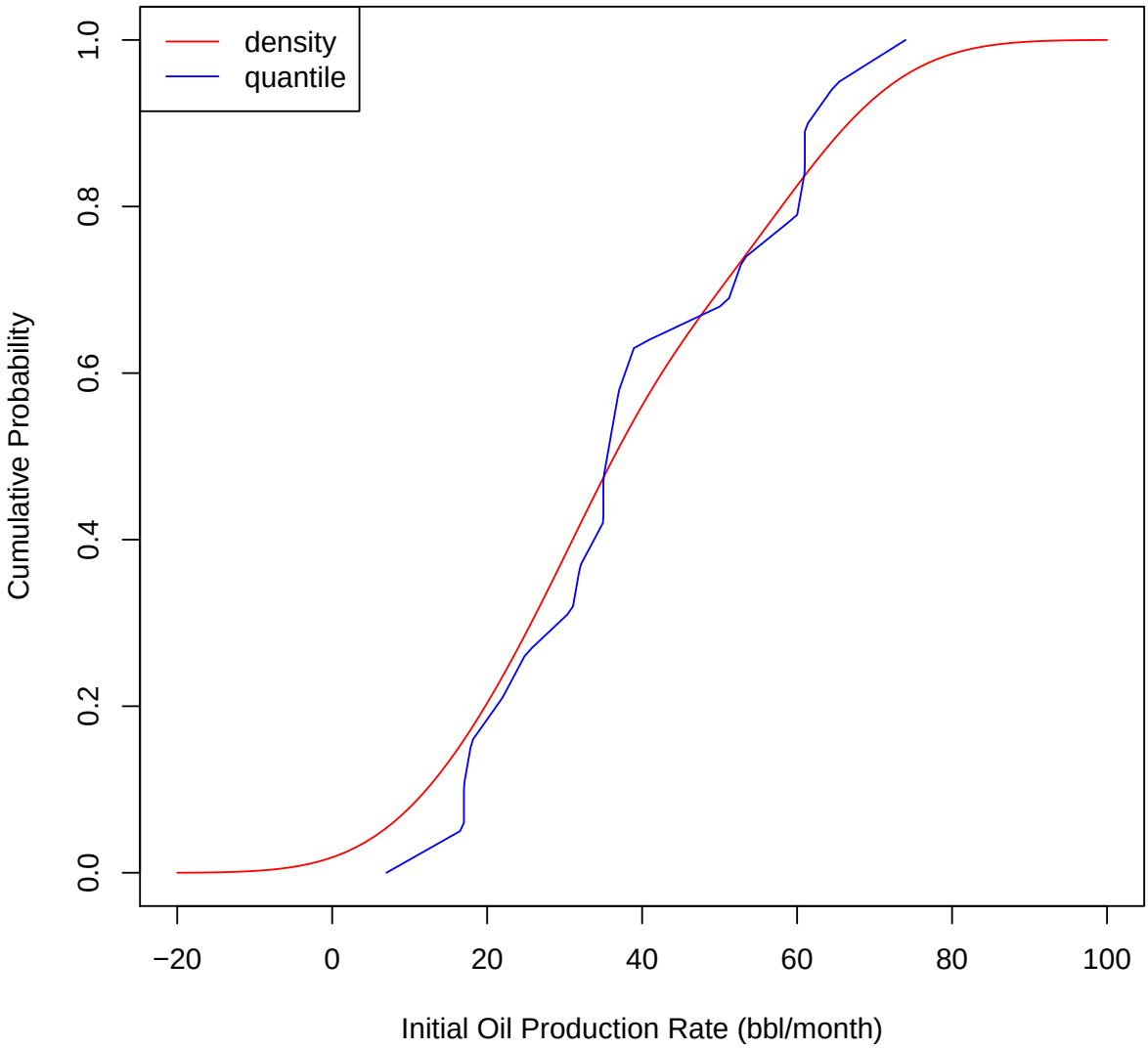

Figure 2.2: Comparison of the CDFs calculated on the basis of the `density` or `quantile` function.

In summary, `CDFd` was the original method used for calculating CDFs for all terms in the model. However we subsequently found that random draws from CDFs generated by `CDFd` did not match the distributions used to create them. `CDFq` was then developed as an alternative method for generating CDFs. `CDFq` limits the CDF values to the minimum and maximum values in the original data set, and reliably reproduces the same distribution used to create it. By default, all CDFs in the model are now generated using `CDFq`, however

`CDFd` has been retained as an alternative method.

2.2.4 `asYear`

This function takes any date object in R and converts it to a numeric year value. For example, if the function was given the following date object (for January 15, 2016, following R's YYYY-MM-DD format for date objects):

`''2016-01-15''`

then the function would return the numeric value `2016`.

2.2.5 `NAoverwrite`

This function finds and replaces any NA values in a vector or matrix object with a specified replacement value (by default the replacement value is 0). It is primarily used to remove NA values from the CPTs used to calculate equipment-based emissions (see Section 3.14), and is intended to prevent the propagation of NA values through that calculation process.

2.2.6 `calTstep`

This function is used to convert calendar dates in the input option files (particularly those in `EF_options`) into the equivalent integer time step value used in the MC simulation using the formula:

$$t_{step} = 1 + round\left[dt(t_{date} - t_{start}) * \frac{12}{365.25}\right] \tag{2.2}$$

where t_{step} is the equivalent MC time step of the calendar date t_{date}, t_{start} is the start date of the MC simulation, dt is the function `difftime` (a built-in function in R for calculating the difference in time between two date objects), and *round* is a rounding function which returns the `difftime` result as an integer. For example, suppose that the MC simulation started on January 1, 2015. This date would therefore be taken as the value for t_{start} and time step 1 of the model. If the user specifies that a pollution reduction for VOC emissions from tanks begins on the month of Jan. 2017 (which would be taken as the value for t_{date}), then `calTstep` would calculate the equivalent time step of that calendar date in the model as:

$$t_{step} = 1 + round\left[(731 days) * \frac{12}{365.25}\right] = 25 \tag{2.3}$$

2.3 Packages

Packages provide additional capabilities and functions not included in the base installation of R. All of the packages used in the model are loaded in Section 1.3 of `main`. A list of the packages utilized and their purpose is shown in Table 2.1. Packages are loaded using the `library` function, however the package must be installed before it can be loaded. R packages can be automatically downloaded and installed via the command:

```
install.packages("[package name]")
```

Replace [package name] with the name of the package you'd like to install. Alternatively, if you are using RStudio you can use the menu option "Tools → Install Packages..." to perform the package installation process.

Table 2.1: Packages utilized by the model and their purpose.

Package	Source	Purpose
foreign	R Core Team [2015b]	Enables use of `read.dbf` function, which allows R to read in UDOGM database files (which are all saved as *.dbf files).
plyr	Wickham [2011]	Used in pre-processing of UDOGM database files, specifically the `ddply` function, to combined the production from different zones of the same well into a single total production record for each well in each month.
zoo	Zeileis and Grothendieck [2005]	Tools for analyzing time series data, such as merging different data sets by a common date index, or calculating monthly/quarterly/annual averages, rolling averages, etc.
data.table	Dowle et al. [2014]	Can be used to create a special kind of data.frame called a data.table, which enables quick indexing and function application. Primarily used in the `dogmDataUpdate` function.
sqldf	Grothendieck [2014]	Allows use of SQL queries in R. Used extensively throughout the data analysis section.
minpack.lm	Elzhov et al. [2013]	Allows use of `nlsLM` function, which is a nonlinear solver that uses the Levenberg-Marquardt algorithm. `nlsLM` is used for fitting all decline curves.
scatterplot3d	Ligges and Mächler [2003]	Used for plotting 3D scatterplots in `postProcess`.
beepr	Baath [2015]	Quality of life feature, enables R to audibly "beep" at the user at the conclusion of the model run.
fitdistrplus	Delignette-Muller and Dutang [2015]	Enables use of the `fitdist` function, for fitting the parameters of known distributions (i.e. Gaussian, gamma, log-normal, etc.) using least squares. Used for projecting the future CDFs of certain decline curve coefficients.
lhs	Carnell [2012]	Latin hypercube sampling (LHS) package for use with `optim` function to pick range of starting parameter values for (3.3) in `drillingModelUpdate` function.
xtable	Dahl [2014]	Used to export R objects as formatted LaTeX tables.
devtools	Wickham and Chang [2016]	Required by openxlsx package (provides necessary zipping tools).
openxlsx	Walker [2015]	Used to export equipment-based emissions results to Excel in `postProcess` script.
forecast	Hyndman [2016]	Provides functions for performing ARIMA fitting and forecasting.
Rtools	R Core Team [2015a]	Rtools is a separate programs required by devtools. It is not a package, and must downloaded and installed from CRAN the same way that R itself is installed.

2.4 Set and Load Options

Finally, the last step of the Global Options section is to set several important options:

- `options(stringsAsFactors = FALSE)`

  By default, R treats character strings in imported text files as "factors." Factors are used in R as categorical variables. For example, several of UDOGM's database files have a column for well type, in which oil wells are specified as "OW" and gas wells as "GW". If this column was typed as a factor, and you subsequently tried to find the mean cumulative oil production from all wells in the database, R would differentiate between oil and gas wells and return the mean of both individually but not collectively, which may or may not be a problem depending on how the rest of your code was setup. In this model we have elected to write the code such that we explicitly define the levels in our analysis, rather than relying on R to implicitly define them.

- `source(''IO_options'')`

  This command loads all of the options set in `IO_options`.

- `source(''EF_options'')`

  This command loads all of the options set in `EF_options`.

- `set.seed(1)`

  This command sets a seed for R's random number generator so that the results of any MC simulation can be reproduced. Random number generation in R always follows a sequence determined by the seed number. By specifying a seed number, randomly generated numbers will always follow the same sequence. Any number can be used, 1 is entered as the default.

- Loading saved results (`if(opt$load.prior == TRUE)` {...})

  If specified by the user, this section of the code loads previously saved model results, cancels further execution of the `main` script, and prints the loading message to the console.

- `writeLines` and `runstart`

  These two commands print a message to the console to inform the user that the data analysis section of the model is about to commence. The start time of the data analysis section is saved as `runstart`, and is subsequently used to calculate the total data analysis run-time.

Chapter 3

Data Analysis

Section 2.x of the model contains all of the data analysis functions in the model. In general, the functions in this segment load an external data source, format and/or restructure that input data, and finally analyze that data to produce a CDF, CPT, or regression model. The updating functions are structured such that `main` will run the function if its update flag in `IO_options` is set to `TRUE`. This allows the user to run the update functions when new data is available, and otherwise to save on model computation time by simply loading the results of the last call to the update function. Each of the functions in Section 2.x are described below.

3.1 `dogmDataUpdate`

This function is used for extracting the information contained in UDOGM's *.dbf database files `proddata`, `welldata`, and `histdata` into a format usable by R (*.rda). As discussed in Chapter 1, these source database files are available for download from UDOGM's Data Research Center website (`https://oilgas.ogm.utah.gov/Data_Center/DataCenter.cfm`) [Utah DOGM, 2015]. In addition to extracting the database files, this function also merges the database files together into a single data.frame called `production` and adds several columns for subsequent analysis. The UDOGM database files contain the following information:

- `proddata`
  - Fourteen columns with one row for every production zone in every well in Utah for every month that well or well zone has been in operation.
  - Contains all of the production records (oil, gas, and water) and number of days each well and/or well zone was in operation.
  - Production records extend from the present back to 1984.
- `welldata`
  - Forty one columns with one row for every well in Utah

  - Contains all well attributes tracked by UDOGM (location, surface lease owner, well type, etc.) and cumulative production volumes for all wells.
  - Information given is the present status of each well. For example, a well might be listed as a water injection well in `welldata`, but that well could have previously been in operation as an oil or gas well.

- `histdata`
  - Thirty one columns with one row for each well event that has been reported to UDOGM.
  - Well events include any event related to work on drilling and well workovers or changes, such as APD approval, spudding, completion, date of first production, well depth, 24 hour production test method and results. etc.
  - This database contains well event records from as early as 1924 to the present, but in most cases well events are only partially reported.

Any wells producing from multiple zones have the production records from all zones summed together so that there is only one total production record from each well in each reporting period. Additionally, since the main item of interest in `histdata` is the original well type, only the first entry for each well in that database is kept in the merged data.frame `production`. Finally, after either updating the database files or loading the saved results from the last time that the `dogmDataUpdate` function was run, the script creates a subset `p` of `production` that contains only those wells located in Uintah or Duchesne counties. Note that this subsetting step occurs outside of the updating function so that both `production` and `p` are available in the model workspace.

3.2 `scheduleUpdate`

This function generates the CDFs for a well's field location, well type, surface lease type (federal, state, tribal, or fee), and well depth. The first step in the function is to create a summary data.frame called `well` with one row for each unique well in `p` with columns for its date of first production, field number, well type, surface lease type, and measured well depth. `well` is then subsetted to selected only those wells that were:

1. Drilled within a specified timeframe.
2. Have a measured well depth greater than or equal to some minimum cutoff.
3. That were either an oil well, gas well, or dry well when they were first drilled (as reported in `histdata`).

One of the key assumptions in our model is that the characteristics of new wells (production rates, depth, surface lease ownership, etc.) will be the same as those of existing wells drilled in the same geographically area. The geographical area that we have elected to use are "fields", which are designations used by UDOGM to distinguish between different oil and gas production regions. Each field in UDOGM's database has a unique name and number (although typically just the field number is used in most database tables). A

list of the top ten largest fields present in the Uinta Basin (based on number of wells and percentage of oil and gas production) are shown in Table 3.1. A full listing of UDOGM recognized fields is available at `https://oilgas.ogm.utah.gov/pub/Publications/Lists/field_list.pdf`. Note that just a small portion of the fields in the Uinta Basin are responsible for the majority of both the drilling and production activity (especially for gas production), which implies that:

1. By analyzing just a few fields individually, we can cover the majority of the oil and gas activity in the Uinta Basin.
2. Since there are very few wells in most fields, there won't be many observations to base CDFs on for most fields. Any CDFs for fields with less than approximately 150 wells are highly likely to have rough and/or discontinuous CDFs (similar to the example CDF shown in Figure 2.2).

Table 3.1: Top 10 largest UDOGM fields by well counts and production (from Jan. 1984 to May 2015).

Field Name	Field #	Oil Wells	Gas Wells	All Wells	Oil (%)	Gas (%)
Natural Buttes	630	73	5436	5509	6.15	68.41
Monument Butte	105	2242	15	2257	15.70	2.81
Brundage Canyon	72	842	21	863	5.28	2.88
Altamont	55	589	19	608	17.76	4.36
8 Mile Flat North	590	419	38	457	2.59	0.90
Bluebell	65	399	8	407	23.71	3.57
Wonsits Valley	710	135	241	376	3.95	2.41
Red Wash	665	230	98	328	4.64	2.00
Antelope Creek	60	324	2	326	2.20	0.39
Pleasant Valley	115	190	1	191	1.22	0.28
All Other Fields	—	1554	952	2506	16.79	11.96

For both of these reasons, we only want to analyze a few fields individually, and any wells located in other fields can be lumped together in a catch-all field category (defined as Field 999). The user input that accomplishes this task is the `field.cutoff` input, which specifies the minimum fraction of wells in the Uinta Basin that a field must have in order to be analyzed individually. `scheduleUpdate` counts the number of wells in each unique field in the `well` data.frame, determines the fraction of wells in each field, selects the fields with fractions greater than or equal to the value of `field.fraction`, and saves out the result as the vector `field`, which is used as an input in many other functions in the data analysis section.

Through trial and error, the best value found for `field.cutoff` (for the purposes of cross-validating against the 2010-2014 time period) was 0.05, which results in analyzing Fields 72, 105, and 630 individually. Field

55 and 65 were excluded (despite their large oil production percentages) because as of Dec. 2009 these fields had only 153 and 161 wells (2% of the total 7,641 wells in the Uinta Basin at that time), respectively. All other fields were either too small to analyze individually or were found to produce unusable CDFs.

Having selected which fields will be analyzed individually (and which will be included in Field 999), we can now calculate the first CDF needed in the model, `cdf.ff`, which gives the probability that a new well will be located in any given field in the Uinta Basin. This is determined the same way that field fractions are calculated, except that only the field fractions in the individually analyzed fields are determined and the difference between their fractions and the total well count in the Uinta Basin are used to find the field fraction for Field 999.

Given that a well is located in a certain field, the next set of probabilities that must be determined is whether a new well is an oil well, gas well, or dry well. The algorithm that accomplishes this counts the number of wells that were initially reported as being: (a) dry, (b) gas, and (c) the total number of wells. From these counts, the function next determines the probability for a well being dry. If the well is not dry, then there is another probability that it will be a gas well, and finally if it isn't a gas well then its an oil well. This process is repeated for each field. These probability picks are cumulative, such that by randomly drawing a number between 0 and 1 and checking which interval that number lies within you can find the well type. For example, suppose that the probabilities were 0.01 of being dry and 0.40 of being a gas well. Any number pick between 0 and 0.01 will be a dry well, between 0.01 and 0.40 a gas well, and greater than 0.40 will be an oil well. In reality, none of the fields analyzed in the Uinta Basin from Jan. 1984 to the present have ever reported a well as being dry initially, but the function provides for that possibility.

The last two factors to be determined in `scheduleUpdate` are surface lease ownership type and well depth. UDOGM identifies four different types of surface lease ownership: federal, Indian, state, and fee (i.e. private land). The probabilities for each type of surface land ownership are calculated off of the `well` data.frame by field. Well depth CDFs are also calculated from `well`, but they are based on well type (oil well or gas well) and not field because there are no clear trends or differences in well depths by field.

3.3 `EIApriceUpdate`

This function loads an externally pre-processed *.csv file with EIA wellhead oil and gas price histories and formats the data for use with other parts of the model. The code in this function is fairly simple, and the only modification made to the input data is an inflation adjustment step that uses the `inf_adj` function (so that the model can potentially use a different CPI index value than the one used in the *.csv file). The result is saved out as a data.frame called `eia.hp`. The actual work in this data analysis step is pre-processing the source data from EIA. A template spreadsheet that accomplishes this task is available at:

`https://docs.google.com/spreadsheets/d/1S1M6RD3QXHewViG-7stioRxxzDDZvSKBHUe4YEVH-CU/edit?usp=sharing`

Monthly UT crude oil first purchase prices (FFP) from July 1977 to present are available from U.S. Energy Information Administration [2015d] at `http://www.eia.gov/dnav/pet/hist/LeafHandler.ashx?n=pet&`

s=f004049__3&f=m, and are entered in the example spreadsheet in column C of the "Data & Calc - Monthly" worksheet. The only change necessary is to adjust these nominal prices for inflation using U.S. Bureau of Labor Statistics [2015] CPI data (entered in column B). Wellhead gas prices are more complicated because there is no single consistent monthly price source for Utah. The data sources used to infer Utah's natural gas wellhead prices are:

1. Utah Natural Gas Wellhead Price [U.S. Energy Information Administration, 2015c]: http://tonto.eia.gov/dnav/ng/hist/na1140_sut_3a.htm","http://tonto.eia.gov/dnav/ng/hist/na1140_sut_3a.htm

   While complete and direct, this data source only gives the annual average wellhead price.

2. U.S. Natural Gas Wellhead Price [U.S. Energy Information Administration, 2015b]: http://www.eia.gov/dnav/ng/hist/n9190us3A.htm","http://www.eia.gov/dnav/ng/hist/n9190us3A.htm

   This source gives monthly wellhead prices, but is currently incomplete (covers time period Jan. 1976 to Dec. 2012) and it's unclear if it will ever be updated.

3. Henry Hub Natural Gas Spot Price [U.S. Energy Information Administration, 2015a]: http://www.eia.gov/dnav/ng/hist/rngwhhda.htm","http://www.eia.gov/dnav/ng/hist/rngwhhda.htm

   The Henry Hub price isn't the first purchase price (FFP) for natural gas, and it isn't local to Utah, but it is the only up to date monthly price list for natural gas available from EIA.

Given the available data sources, the process for calculating natural gas wellhead prices in Utah is as follows:

1. Find the average ratio of (Utah)/(U.S.) natural gas wellhead prices on an annual basis.
2. Multiply all monthly U.S. natural gas wellhead prices by that ratio to find the Utah wellhead price.
3. Once the U.S. natural gas wellhead price data ends, use the average ratio of (Henry Hub price)/(U.S. wellhead price) to find the U.S. wellhead price after Dec. 2012.

The final result is given on the worksheet labeled "CSV Export - Monthly". The data on that worksheet should be exported to a *.csv file labeled "EIA_HistPrices.csv" located in the "raw" data filepath.

3.4 leaseOpCostUpdate

This function fits a least squares model to lease equipment and operating cost estimates published by EIA for domestic oil and gas operations between 1994 - 2009 [U.S. Energy Information Administration, 2010]. The source data is contained in multiple tabs of an Excel spreadsheet, which were reformatted into a *.csv file with the following columns: year, consumer price index (CPI), nominal price (nominal $/bbl), real price (2009 $/bbl), well depth (ft), production rate (for gas only, units of MCF/day or MCFD), and lease operating costs (2009 $/year). The original costs in 2009 dollars are adjusted for inflation to a 2014 dollar basis and then the data is fit with the following equations for both equipment and operating costs:

$$LC_{oil} = aOP + bD + c \tag{3.1}$$

$$LC_{gas} = aGP + bD + cPR + d \tag{3.2}$$

where LC_{oil} and LC_{gas} are the lease equipment or operating costs for oil and gas wells, respectively, OP is oil price (in 2014 \$/bbl), GP is gas price (in 2014 \$/MCF), D is well depth in feet, PR is gas production rate (in MCFD), and all other variables are fitted coefficients.

Since this report has been discontinued by EIA, there is likely no need to revise or change any of the inputs to this function in the future.

3.5 `drillingModelUpdate`

The number of wells drilled each month in the Uinta Basin can be modeled as a function of past and present energy prices. In economics, these kind of functions are referred to as distributed-lag models. This function fits each of the following forms of distributed-lag models to the drilling history in the Uinta Basin:

$$W_t = aOP_t + bGP_t + cW_{t-1} + d \tag{3.3}$$

$$W_t = aOP_{t-1} + bGP_{t-N} + c \tag{3.4}$$

$$W_t = aOP_{t-N} + b \tag{3.5}$$

$$W_t = aGP_{t-N} + b \tag{3.6}$$

In the equations above W is the number of wells drilled (oil, gas, or dry), t is the time step (in months), OP is the FPP of oil in Utah (\$/bbl), GP is the FPP of gas in Utah, and all other terms are fitted coefficients. The `drillingModelUpdate` function fits each of the above equations simultaneously over the specified time period so that the user can later choose between any of the models in the MC simulation. Note that the fitted coefficient N is an integer value. Equations (3.4)-(3.6) are fitted for all values of N from 1 to the maximum time-lag value specified in `IO_options`, and the value of N that produces the smallest residual error is automatically selected for subsequent use in the model. Equation (3.3) uses a fixed value of $N = 1$ for the prior well term because doing so effectively applies exponential smoothing to the entire price history up to the time step t as discussed in Lozada and Hogue [2008].

Equations (3.4)-(3.6) all use R's built in linear regression function `lm`. However since Equation (3.3) involves

the term for the number of wells drilled in the prior time step (W_{t-1}), it can't be solved using `lm` because the value of W_{t-1} is dependent on the fitted coefficients. As a result, the optimization function `optim` is used to find the set of fitted coefficients that minimize the residual sum of the squares (RSS) error between the estimate and the actual drilling schedule. Since `optim` requires (and is highly sensitive to) the initial guesses supplied to it, a Latin hypercube sample (LHS) is drawn using the package `lhs` to pick a representative range of possible starting guesses. The value ranges for the initial guesses are hard-coded into the `drillingModelUpdate` function, but the range is wide enough that it shouldn't need to be modified.

3.6 `ARIMAfitUpdate`

This function fits both a manually specified and automatically selected auto-regressive integrated moving average (ARIMA) model to the past oil and gas price histories in the Uinta Basin for subsequent use in generating long-term price forecasts.

ARIMA models are a commonly used type of time-series forecast. In a time-series forecast, the future values of some quantity are predicted solely based on the past values of that same quantity. In general, the quantity to be forecasted is assumed to be comprised of (a) seasonal variation, (b) an underlying trend, and (c) noise. The seasonal decomposition of the oil and gas price histories for in the Uinta Basin are shown in Figures 3.1 and 3.2.

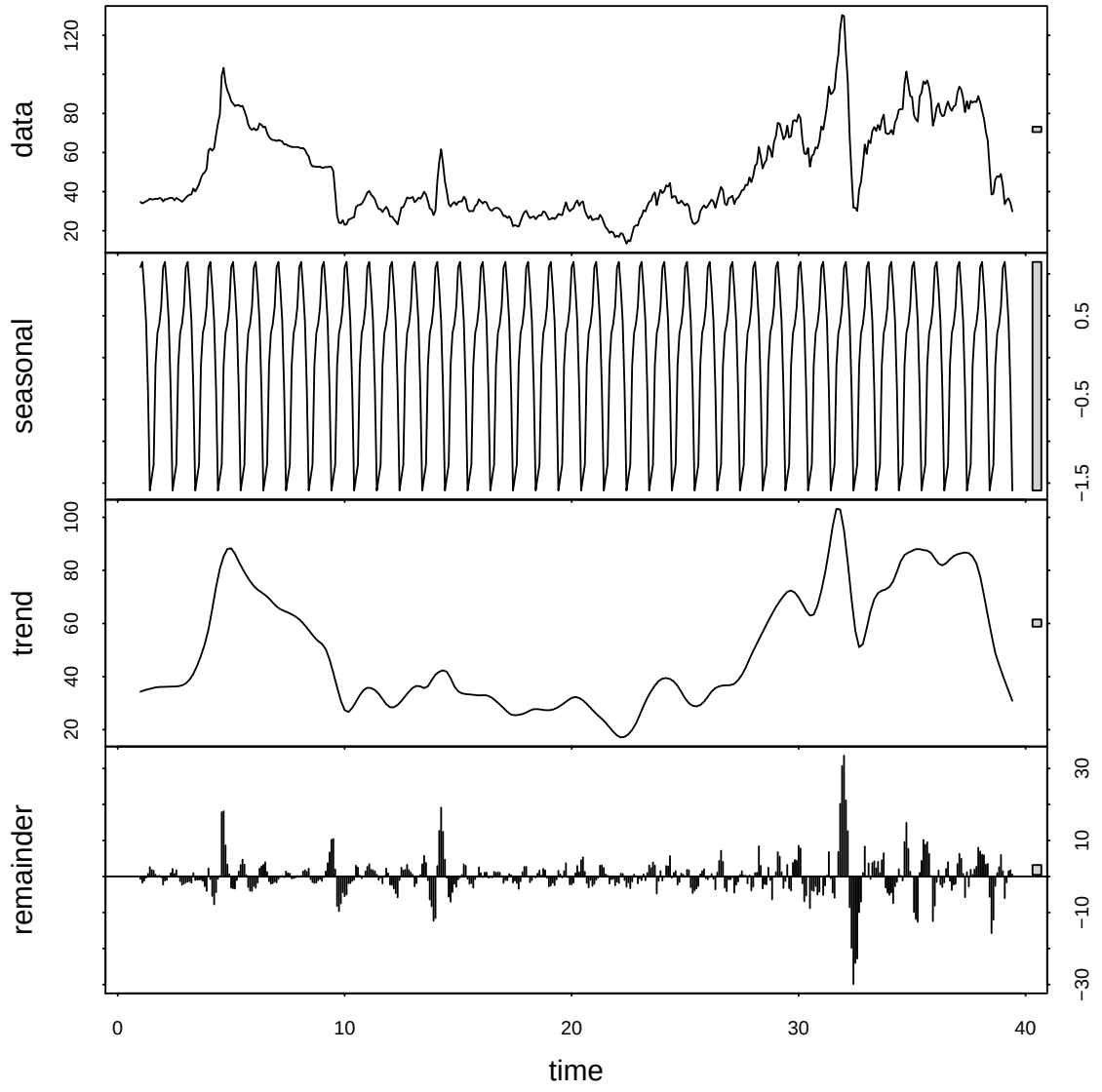

Figure 3.1: Time-series decomposition of oil prices ($ / bbl) in the Uinta Basin. Note that the "time" values on the x-axis are years since 1977.

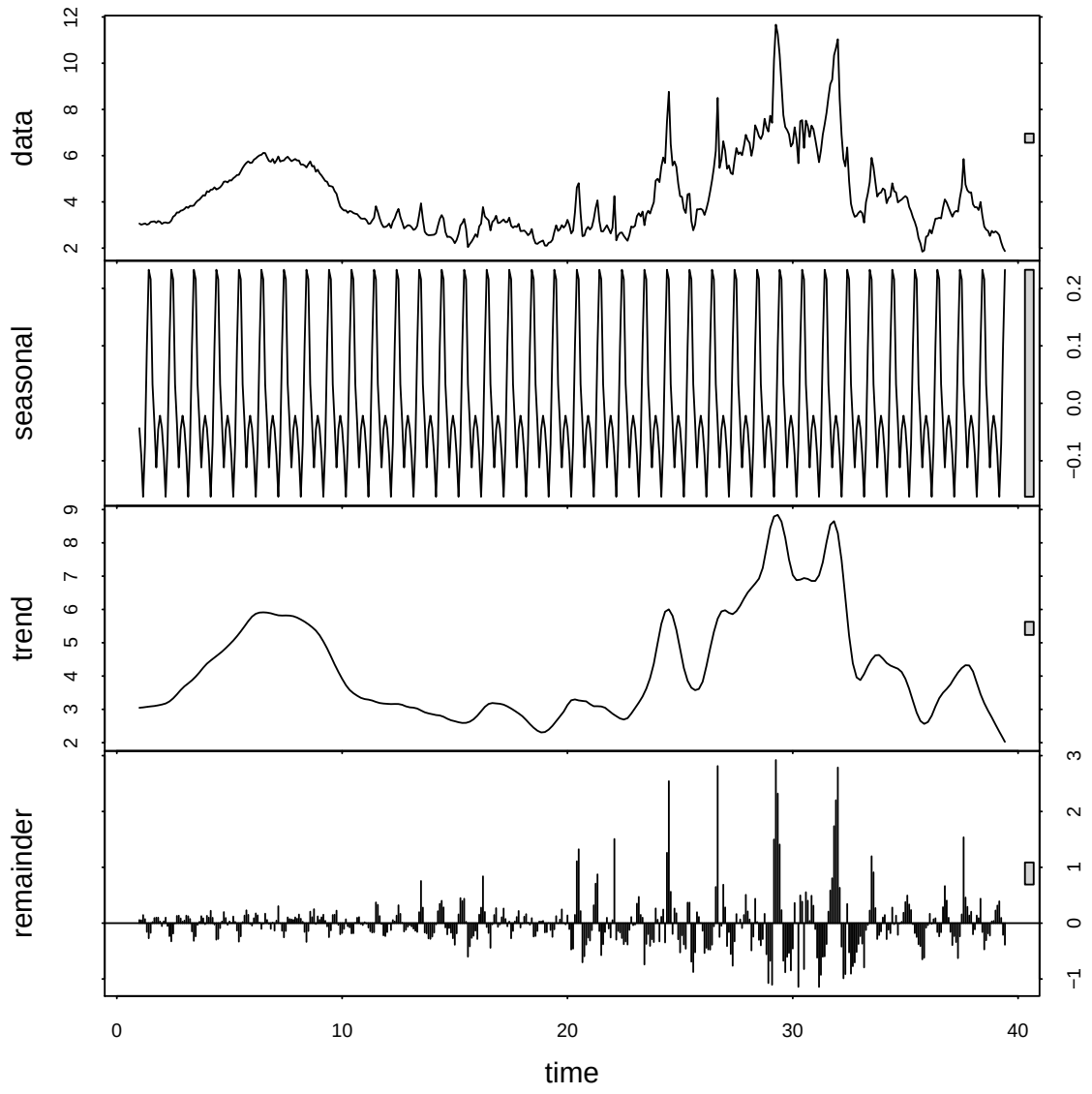

Figure 3.2: Time-series decomposition of gas prices ($ / MCF) in the Uinta Basin. Note that the "time" values on the x-axis are years since 1977.

From Figures 3.1 and 3.2, we can conclude the following:

1. Seasonal variation in oil prices (±$1.5 / bbl) and natural gas prices (±$0.2 / MCF) is negligible.
2. Over their entire price history, the trend for both oil and gas is flat. There are clearly periods where prices went up during the early 1980s and mid 2000s, but long-term there is no clear upward/downward trajectory.

3. There is substantial noise in the remainder term (±$30 / bbl for oil and +$3 -$1 / MCF for gas).

These features inform what type of ARIMA model to use for making long-term forecasts. In an ARIMA model, the underlying trend is fitted using the past values (p), differences in past values (d), and the moving average of the fitting error (q) as predictor variables. ARIMA models are classified based on (a) the order of each type of these predictor variables (p,d,q) they use, and (b) whether or not they include a constant term. Certain ARIMA models are known to have characteristic behaviors (e.g. they decay to zero, result in a random walk around a constant mean, etc.). Several noteable types of ARIMA(p,d,q) models are discussed below:

1. (0,0,0) with or without a constant - White noise

   (Not recommended) Produces random values that have the same distribution of values as the data used to fit the model. Quantiles of energy price forecasts produced with this model will appear to be constant over time.

2. (1,0,0) with or without a constant - First-order auto-regressive model

   (Not recommended) If a time-series is "stationary" (i.e. it has no trend, variations around the mean are all of the same magnitude, and in general none of its statistical properites vary over time), then future values can be predicted as a multiple of its own previous value, plus a constant. If the multiplier is positive and less than 1, the model will be mean-reverting. This type of model is not recommended because the oil and gas price histories in the Uinta Basin are not stationary.

3. (0,1,0) without a constant - Random walk

   Starting from an initial value, the price will randomly step up or down at each price step. The resulting model will produce forecasts that have a constant median equal to the starting initial value. This type of model is equivalent to the previously used Geometric Brownian Motion model.

4. (0,1,0) with a constant - Random walk with drift

   Same as the previous model, but adding a constant term will fit a straight line through the training data and extrapolate that line over the entire forecast period as the median forecasted price (with random variation around the median).

5. (0,1,1) without a constant - Exponential smoothing

   (Recommended model) Uses an exponentially weighted moving-average of the error in past prediction values in addition to the difference term to forecast each future values. This type of model is best for modeling time-series with noisy fluctuations around a slowly-varying mean (which accurately describes the oil and gas price histories in Figures 3.1 and 3.2).

6. (0,1,1) with a constant - Exponential smoothing with growth

   Including a constant in a (0,1,1) model adds a trend to the forecast (in the same way as the (0,1,0) with a constant does).

Unless the user believes there is reason to project a price trend with the (0,1,0) or (0,1,1) models, it's best to use the ARIMA models without a constant.

Returning to the implementation of how the ARIMA models are fit in the `ARIMAfitUpdate` function, the user must specify a "manually" selected model (e.g. (0,1,1) without a constant). This manually specified model is fit using functions in the forecast package. Additionally, the `auto.arima` function from the forecast package is used to find the best fitting ARIMA model for both oil and gas prices as measured by information criteria scores (e.g. Akaike information criterion, Bayesian information criterion, etc.). Either ARIMA fit (manual or automatic) can be used subsequently to make predictions. The user is advised to review the plots generated by the `ARIMAfitUpdate` function to compare the performance of both model types.

3.7 EIAforecastUpdate

This function takes the Annual Energy Outlook (AEO) forecast data entered in `IO_options` and adjusts it for (1) inflation (using the CPI), and then (2) interpolates monthly oil/gas prices from the annual forecast values. The interpolation is linear and assumes that the EIA AEO forecast prices occur during the middle of each calendar year (i.e. June). Three EIA forecasts are considered in the model: the reference case (used to generate the energy price forecast in the MC simulation), and a low/high case (which are included for reference in `postProcess`'s predefined energy price forecast plots).

3.8 EIAerrorUpdate

This function generates the CDFs for the relative error in EIA's Annual Energy Outlook (AEO) reference oil and gas forecasts as a function of how far into the future the forecast is predicting. The source data for generating these CDFs comes from EIA's AEO Reports from 1999 - present [U.S. Energy Information Administration, 2014] (using only EIA's reported Rocky Mountain region wellhead oil and gas prices), and the actual average annual wellhead energy prices from `eia.hp`. Relative error (RE) is defined here as:

$$RE = \frac{(PP - AP)}{PP} \tag{3.7}$$

Where PP is the EIA price prediction and AP is the actual price. Note that this method for calculating relative error is no longer preferred and has been replaced by the alternative method discussed in Section 3.9. However `EIAerrorUpdate` has been retained in the model in case the user wishes to use this method for propagating EIA's forecasting error.

This function also requires a fair amount of pre-processing outside of R in a spreadsheet. A template spreadsheet is available at:

`https://docs.google.com/spreadsheets/d/1kew_vlpULLMUyWXxeD70HRnsO83ebCHfHMjaCsUrxYc/edit?usp=sharing`

To update the spreadsheet, follow the general steps listed below (steps are for updating the oil relative error *.csv file, but the steps are identical for gas):

1. Calculate new annual average wellhead oil prices from `eia.hp`. Add these values to the bottom of the first table on the "EIA Oil FC" worksheet.

2. For every year for which new actual prices are available, fill in the AEO forecasted price for each AEO report. Copies of each report are available from EIA at:

   `http://www.eia.gov/forecasts/aeo/archive.cfm`.

   Note that the prices are for the Rocky Mountain region, and that all prices must be inflation adjusted to the same index value (for example, all costs in the template are adjusted to 2014 USD using the CPI).

3. The second table contains the relative error calculation formulas. Extend the second table to match the dimensions of the first table, moving any fixed references as necessary.

4. Repeat with the third table at the bottom of the worksheet, which presents the same information but shifted so that the error in each AEO forecast is normalized by the number of years into the future that the forecast is being made. Copy and paste the values from this last table (without the column or row headers) into the "op_error_export" worksheet.

5. Download the export worksheet as a *.csv file. Make sure that the file name matches the one used in `EIAerrorUpdate`, which by default is "EIA_op_error_export.csv". Finally, place the *.csv file in your "raw" data folder path.

The actual code in the function begins by loading the *.csv files containing the percent relative error data. These files are converted into a data.frame, and R's normal CDF function `qnorm` is used to generate the CDF for the relative error for each column (i.e. the number of years into the future being predicted), under the assumption that the relative errors can be modeled as a normal distribution. The resulting CDF matrices `Eoil` and `Egas` are then saved.

3.9 EIAerrorUpdateFrac

This function calculates the relative error (RE) between the actual price (AP) of oil and gas in Utah and EIA's forecasted prices (FP) using the equation:

$$RE = \frac{FP}{AP} \tag{3.8}$$

if $FP < AP$, or:

$$RE = \frac{AP}{FP} \tag{3.9}$$

if $FP > AP$. Defining RE in this way is useful because:

1. The value of RE is always bounded between 0 and 1 and can be described using a beta probability distribution.

2. It captures the absolute magnitude of the relative error. While EIA under-predicted actual prices for oil over the 1999-2007 time period, forecasts from 2009 onwards have over-predicted actual prices for oil (gas prices have followed a similar pattern). There is no evidence that EIA's forecasts are systemically under- or over-predicting energy prices.

3. Equations 3.8 and 3.9 avoid a mathematical pitfall that occurs with the absolute value calculation of RE in Equation 3.7. Substituting the simulated price (SP) for AP in Equation 3.7 and solving for SP gives:

$$SP = FP * (1 + RE) \tag{3.10}$$

   If a negative value of RE is selected during the MC simulation process (entirely possible given the range of RE values calculated by Equation 3.7), then SP may be negative, which is not a realistic result. By comparison, solving Equations 3.8 and 3.9 always returns a result bounded between [0, ∞]:

$$SP = \frac{FP}{RE} \tag{3.11}$$

$$SP = RE * FP \tag{3.12}$$

The pre-processing steps for the EIA source data are very similar to those outlined for `EIAerrorUpdate` in Section 3.8 (and the template spreadsheet referenced in Section 3.8 contains worksheets for both types of RE analysis). The steps to follow are (identical for both oil and gas):

1. Calculate new annual average wellhead oil prices from `eia.hp`. Add these values to the bottom of the first table on the "EIA Oil FC fraction" worksheet.

2. For every year for which new actual prices are available, fill in the AEO forecasted price for each AEO report. These will be identical to the values used on the "EIA Oil FC" worksheet.

3. The second table contains the relative error calculation formulas (using Equations 3.8 and 3.9). Extend the second table to match the dimensions of the first table, moving any fixed references as necessary.

4. Repeat with the third table at the bottom of the worksheet, which presents the same information but shifted so that the error in each AEO forecast is normalized by the number of years into the future that the forecast is being made. Copy and paste the values from this last table (without the column or row headers) into the "op_fraction_export" worksheet.

5. Reformat the RE values into two columns containing (a) a numerical value indicating the number of years into the future associated with the given RE value, and (b) the RE value itself.

6. Download the export worksheet as a *.csv file. The file name must match the name used in the script, which by default is "EIA AEO frac error op export.csv". Finally, place the *.csv file in your "raw" data folder path.

The actual code in the function begins by loading the *.csv files containing the relative error data. The data set is converted into a data.frame and a beta distribution is fitted to the RE values for each year. The fitted parameters are used to generate a CDF error matrix for oil (`EoilFrac`) and gas (`EgasFrac`) that follow the same format as the `Eoil` and `Egas` matrices discussed in Section 3.8. Finally, `EoilFrac` and `EgasFrac` are saved to disk.

3.10 `DCAupdate`

This function performs a well-by-well decline curve analysis (DCA) to enable the prediction of future production rates from existing wells, and to generate the dataset necessary to estimate the production from future wells. In general, the rate of production of oil and gas declines over time. Numerous decline curve equations have been developed for different specific oil and gas reservoir conditions. The two forms of decline curve equations used here are the hyperbolic decline curve equation (Equation (3.13), from Arps [1945]), and the cumulative production equation (Equation (3.14), from Walton [2014]):

$$q(t) = q_i(1 + bD_i t)^{-\frac{1}{b}} \tag{3.13}$$

$$Q(t) = C_p\sqrt{t} + c_1 \tag{3.14}$$

In Equation (3.13) q is the production rate at time t, q_i is the initial production rate, D_i is the initial decline rate, and b is the decline exponent. In Equation (3.14) Q is the cumulative production at time t, and C_p and c_1 are fitted coefficients.

The actual decline curve analysis method consists of applying Equation (3.13) or (3.14) to the production records of a given well and finding the best fit by least-squares over a given time period. However this task is significantly complicated by the tumultuous production records of many wells, which are frequently noisy and which may contain "late starts" and/or "restarts." Late starts are wells that initially come online at a relatively low production rate, and then later reach a much higher production rate from which the curve begins its true decline. Restarts are any event which leads to a major increase in well production rates, followed by a subsequent but separate decline curve. In either case, the nonlinear solver typically fails to converge or finds terrible fits for these type of production records.

The workaround then is to identify the actual start/stop points of each decline curve segment in each well's production record. The algorithm for accomplishing this task (and for implementing a number of other fixes to problems that can occur) is described below:

1. Subset the data.frame `p` to only include production records from oil or gas wells within the time range of interest and which produced for at least a minimum number of days per month.

   There are two things to note about this step. First, the requirement that the wells produce for at least a minimum number of days per month substantially cuts down on noise by removing from the dataset

production records that don't accurately reflect a full month of production.

Second, regarding time ranges, there are two ranges of interest. If the model is being cross-validated, then some portion of the production dataset must be excluded from the DCA for testing the model's predictions. However for both decline curve distribution fitting (discussed later) and for full prediction modes, it's desirable to fit with all of the production records available. Since having both the full and time-constrained results in the model workspace at the same may be necessary, the `DCAupdate` function is set to run both with and without time subsetting constraints.

2. Get a list of unique wells in the subsetted production data and determine how "old" each well is at the start of the MC simulation period. Age here is an indication of how far along into its production decline curve each individual well is, so that production from existing wells can be calculated for each individual well based on its age.

3. For each well, and for both gas and oil:

   (a) Select the production records for the given well and product type, and further select only the records which are > 0. Note that there are actually records which are negative in the UDOGM database. Another reason for the > 0 requirement is that even if a well is no longer a producing (e.g. it was converted into a water injection well, shut-in, etc.), it will still have a zero production record for every month of its existence.

   (b) Check if the number of remaining production records is greater than some minimum number of production records. This step is intended to prevent over-fitting newly drilled wells which might only have a couple of data points. If there are too few records, skip fitting the well, otherwise move onto the next step.

   (c) Determine the start and stop dates for each unique decline curve in the given well's production record. This is done by summing production records together over a specified time window (e.g. 12 months) into "bins". If a well is constantly declining, then the difference between values of adjacent bins will always be negative. The function checks this by first normalizing the bin values and then calculating the differences between each bin. If any bin has a positive difference value above a minimum specified threshold (e.g. a positive relative increase of $> 15\%$), then the preceding bin must contain the endpoint of a decline curve. The the preceding bin is searched for its minimum value, and the point in time at which that record occurs is the stop point of the decline curve. The search for stop points repeats for the entire production record, and then the start points are found by searching for the maximum production record value between any stop points (or the start of the production record).

   There are a number of nuances to the binning process. First, the size of the bin time window and the minimum positive difference value both have a big impact on identifying the start/stop points. Large bin time windows smooth out noisy production curves and are more likely to find the true minimum point at the end of each decline curve. Conversely, the smoothing effect also means that large bins are less sensitive and are more likely to miss rapidly changing trends in the production records. The minimum positive difference value is a judgment call that is dependent on the choice of binning window. Too low (for the given bin size) and you will produce false positives; too high and clear decline curve stop points will be missed.

Secondly, searching for start points isn't always as simple as picking the maximum value. We found several instances where the maximum production value was an outlier that occurred near the end of a decline curve segment. To act as a check against this possibility, the search for the maximum production value is actually extended to the top n production values. The earliest occuring value in the set of n values is then picked as the start point.

Lastly, the early segment of most production records is highly volatile. To avoid getting too many false positives in this early period, another parameter was added which specifies how many bins into the production record a potential stop point must be in order to be considered an actual stop point. Any stop points found in the early segment are ignored.

(d) Given the selected start/stop points, calculate the time difference between when the well first went online and when the start of the first decline curve occurred. This difference is the "time delay," and it is used to estimate the time between when a new well is drilled and when it first begins production.

(e) Run the nonlinear solver on each decline curve segment using both forms of the decline curve equation. Depending on how the decline curve segment search went, there will be either one continuous curve or multiple curves. Intermediate curves are ignored, and only the first curve (used for estimating production from new wells) and last curve (used for extrapolating production from existing wells) are fitted.

It should be noted that the solver checks again that the length of each decline curve segment is at least equal to the minimum number of production records, skipping the segment if the requirement isn't met.

There are many inputs for the nonlinear solver. Each coefficient in Equations (3.13) and (3.14) requires an initial guess and a set of upper and lower limit values (for both oil and gas). The majority of the inputs to `DCAupdate` are for the nonlinear solver.

(f) If specified by the user, the function can plot the production record and fit results for each well. Plots are grouped together by field and by product type and are saved as *.pdf files in the "plots" file path.

4. Finally, rearrange the fit results into the data.frames `mo` (for oil curves) and `mg` (for gas curves) and save. Each row of these data.frames represents a unique well, and contains columns summarizing the hyperbolic and cumulative curve fits for both the first and last curve of each well. `mo` and `mg` also contain a set of flags indicating the result of the fitting procedure (fitted, skipped, or failed).

3.11 `DCAupdateCDF` and `QfitDCAupdateCDF`

This function takes the data.frame of fit results from `DCAupdate` and generates a CDF for each coefficient in Equations (3.13) and (3.14) for each field being analyzed. The function begins by selecting a subset of the fit results to ensure that the CDF reflects a realistic range of values (i.e. eliminates fitting outliers) and only including fit results for wells drilled within the specified time range. The function supports estimating the CDF for the selected data points using either the `density` or `quantile` functions. However as discussed previously, the `quantile` method is preferred.

3.12 reworkUpdate

This function calculates the CDF that describes the probability of a well being reworked (i.e. reperforated or recompleted) as a function of how far a well is into its productive life and what type of well it is (oil or gas). The function begins by loading the `histdata` data.frame from disk. Next, the function selects from `histdata` only the wells located in the Uinta Basin. The `histdata` information is further subsetted into one data.frame containing the dates when a well was drilled during the time frame of interest, and a second data.frame containing the dates when a well was reworked. The number of wells that have existed for at least n months is calculated from month 1 to the age of the oldest well in the dataset. Next, the number of wells that were reworked n months into their productive life is calculated. The rework vector is divided by the vector containing the number of wells present to get the probability that a well will be reworked as a function of time, and that probability is summed together to create the CDF for reworking. Note that at some point in time the number of wells remaining in the data set that are at least n months old will become too small to draw any statistically meaningful relationship between well age and the rework probability. Therefore the `reworkUpdate` function includes an input option to stop estimating the rework CDF if there are fewer than some minimum number of wells left in the data set (100 wells by default).

3.13 DCAlnormUpdate

This function was created to (1) fit a log-normal distribution to the coefficients C_p and c_1 in Equation (3.14), and then (2) perform a linear regression on the fitted log-normal distribution parameters (the log-mean and log-standard deviation). The resulting linear regression can then be used to project the log-normal distribution parameters for cumulative oil or gas production for new wells. This approach was developed to counter a problem encountered with predicting oil production from new wells. If the CDF for C_p and c_1 calculated by `QfitDCAupdateCDF` is used to predict oil production from new wells during the 2010-2014 time period based on DCA fits from the 1984-2009 time period, the model significantly under-predicts oil production, even if the exact drilling schedule is taken as a given as shown in Figure 3.3.

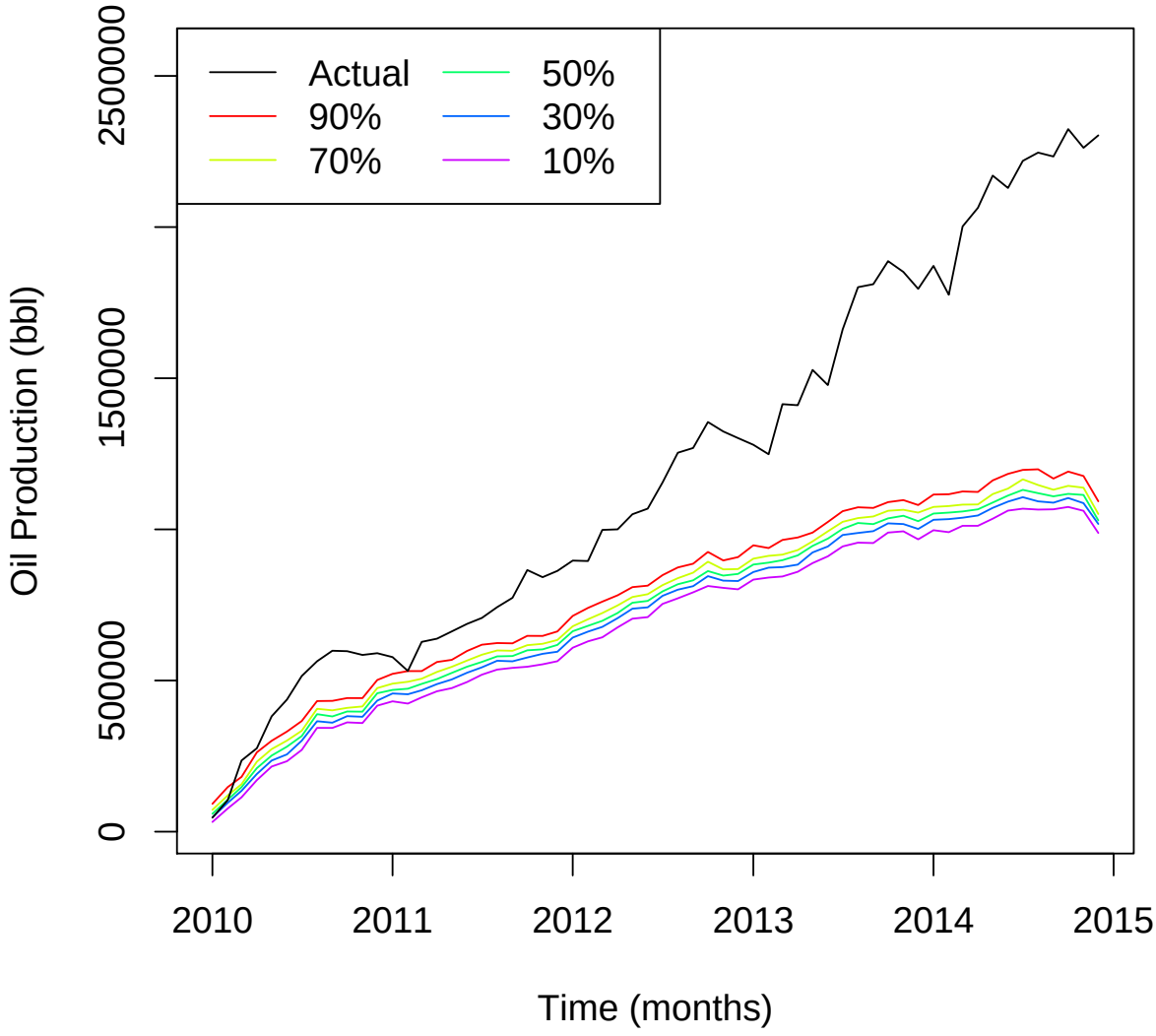

Figure 3.3: Cross-validation comparison of oil production from new wells using CDFs from `QfitDCAupdateCDF`. The time period 1984-2009 was used as the training period, and 2010-2014 as the test period. The exact drilling schedule for new wells was taken as a given.

The reason for the discrepancy is that the production rates from wells drilled during the 2010-2014 time period increased signficantly compared to wells drilled during the test period. Figure 3.4 shows the CDF for C_p for oil wells by year (labeled in the figure as years since 1984). The dotted-lines, equivalent to the CDF for C_p from oil wells drilled during the test period, clearly show this shift in production rates. Since the training period misses this shift in the C_p values for oil wells, there's no way to match the production rates for new wells using solely the empirical CDFs for C_p calculated in the `QfitDCAupdateCDF` function.

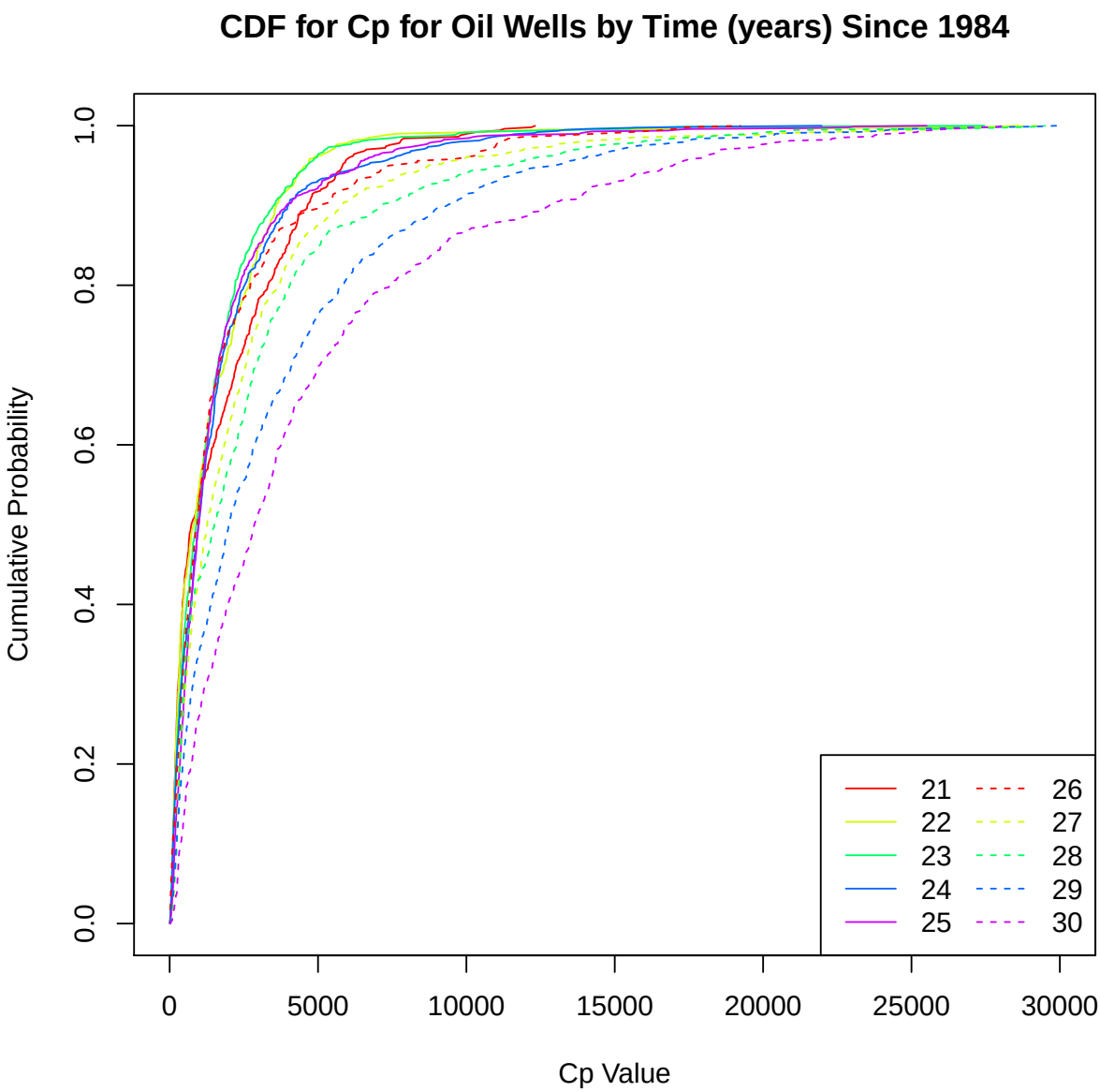

Figure 3.4: CDFs for C_p for oil wells by year. Years in the figure are shown as years since 1984. CDFs drawn with dotted lines are equivalent to the CDFs for wells drilled during the test period 2010-2014.

The two step approach in this function (fitting the coefficients in Equation (3.14) with a known distribution and then projecting the distribution parameters into the future using linear regression) can fix this problem if the selected regression time period is a match for future production. Figure 3.5 shows the linear regression results for fits of C_p and c_1 log-normal distribution parameters at the Basin-scale (i.e. for all wells in the Uinta Basin).

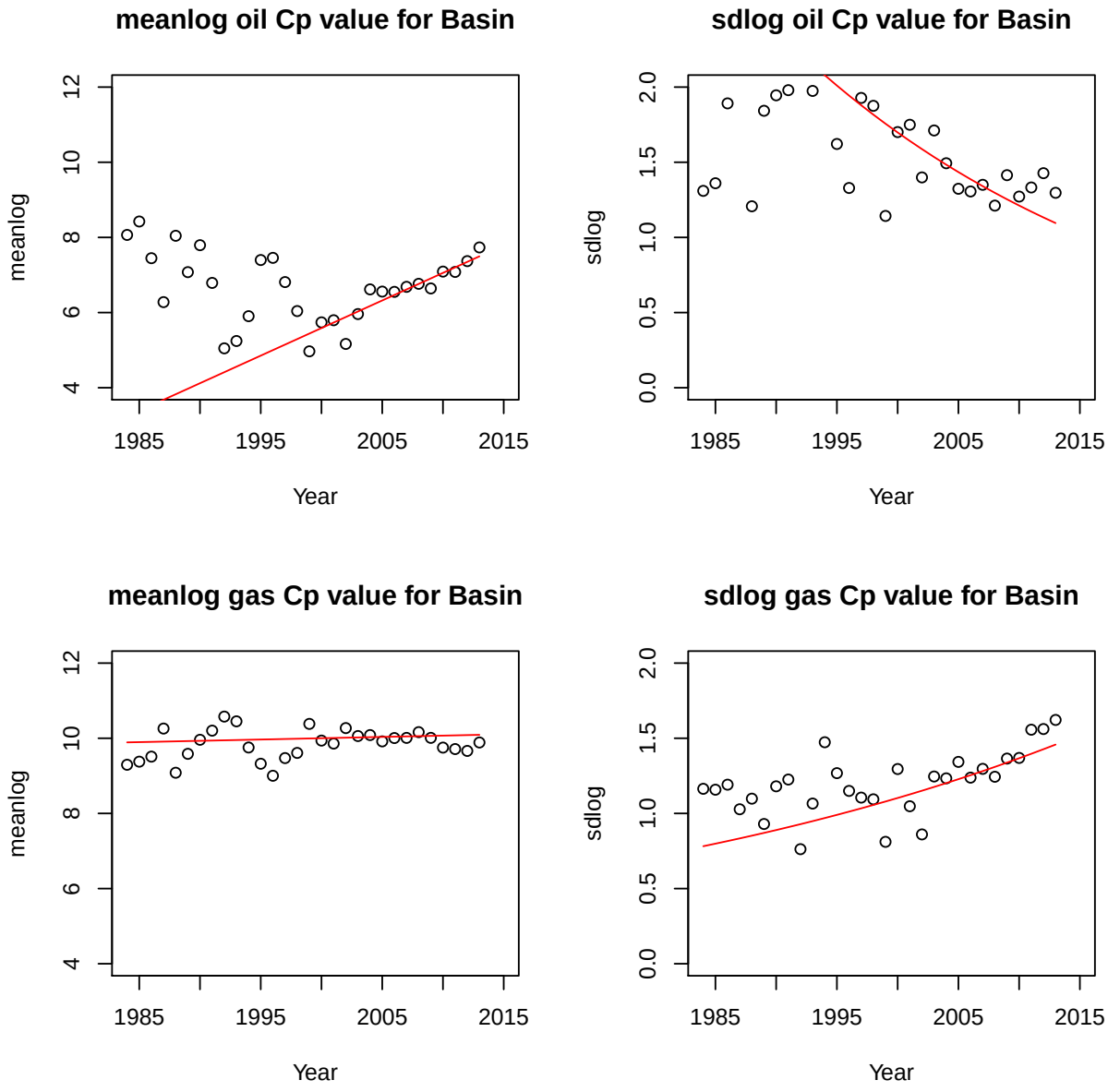

Figure 3.5: Linear regression fits of C_p and c_1 log-normal distribution parameters for all wells in the Uinta Basin. The data points used for fitting each trendline were from years 1999-2009 (with projections to other years shown).

Various time periods were selected for fitting and projecting distribution parameters, and the overall best fit to the production during the cross-validation period was achieved by fitting to data points in the 1999-2009 time period. The results of predicting oil production from new wells with the distribution parameter projection shown in Figure 3.4 are given in Figure 3.6 (again, taking the actual drilling schedule as a given). Figure 3.6 is an excellent match to the actual production rates during that the 2010-2014 cross-validation period, and if the trend of increasing C_p values for oil wells continues then this method should give the most

accurate method for predicting future oil production from new wells in the Uinta Basin.

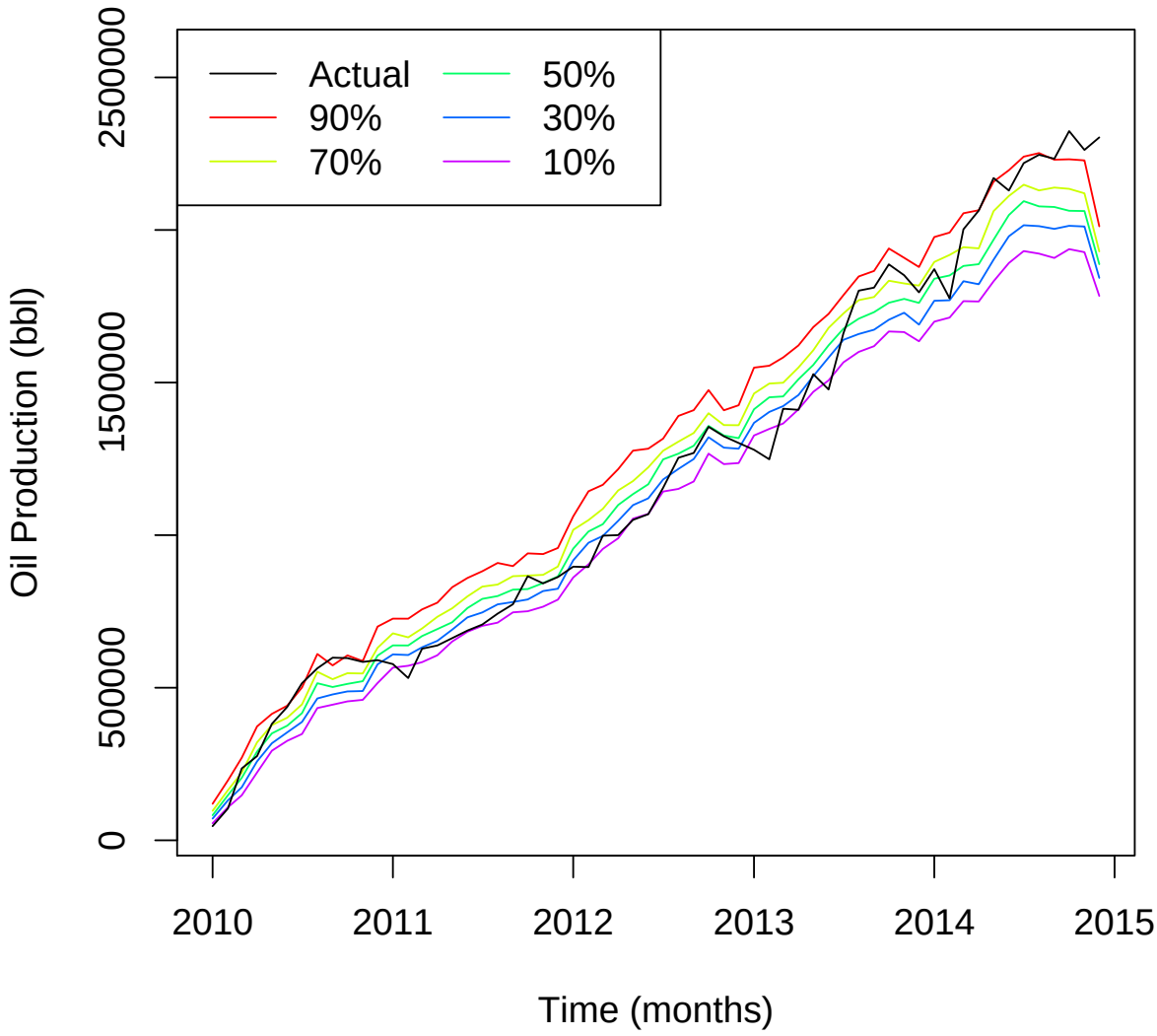

Figure 3.6: Cross-validation comparison of oil production from new wells using log-normal distribution CDFs for C_p as projected by the distribution parameter fit from `DCAlnormUpdate`. The time period 1999-2009 was used as the linear regression period for distribution parameter fitting, and 2010-2014 was used as the test period. The exact drilling schedule for new wells was taken as a given.

3.14 emissionUpdate

This function finds all of the unique combinations of inputs required for performing equipment-based emissions calculations using the data contained in the oil and gas emissions inventory (OGEI) database and returns the result as a cumulative probability table (CPT). CPTs are created by:

1. Selecting the columns containing inputs variables needed for performing the equipment-based emissions calculations from each OGEI database table.
2. Find all unique combinations of the selected input variables.
3. Count the number or times each unique combination appears in the table.
4. Calculate the cumulative probability for each combination.

For example, if the input variables in a table were:

Input 1	Input 2
a	1
a	1
a	1
b	2
b	1
c	3

Then the CPT for that table would be:

Input 1	Input 2	Count	Fraction	Cumulative Probability
a	1	3	0.500	0.500
b	2	1	0.167	0.667
b	1	1	0.167	0.833
c	3	1	0.167	1.000

The motivation for using a CPT is two-fold. First, CPTs fit with the rest of the MC simulation's structure (randomly picking a number between 0 and 1 and finding the corresponding value from a table or CDF). Second, CPTs safeguard against independently picking values for input variables that should be correlated. As an example of this second point, consider the emission factors reported by operators for RICE and

turbines, which can be any combination of AP 42, NSPS JJJJ, stack test, or manufacturer specification (for each pollutant category). If each reported emission factor were described using a CDF, then during simulations random selections from each CDF might produce combinations that are never observed in the OGEI database (e.g. a low $PM_{2.5}$ but high PM_{10} emission factor). Given the differences in emission factors between each unique RICE or turbine, treating all inputs for that equipment category in combination is a more representative approach.

The steps followed to create each CPTs are:

1. Export OGEI database tables to *.csv files.

   In order to analyze the files contained in the OGEI database, the user must export the following tables must from the OGEI database as csv files (using SqliteBrowser, or any similar program):

Table 3.2: List of tables that must be exported from OGEI database as *.csv files. Exported files should use same name as table that they are representing (e.g. "apis" table should be named "apis.csv").

Table	Contents
apis	API numbers associated with each facility ID
component_count	Component counts
dehydrators	Dehydrators
facilities_list	Facilities list
fugitives	Fugitive emissions
operator_info	Operator information
pneumatic_controllers	Pneumatic controllers
pneumatic_pumps	Pneumatic pumps
production_areas	Production areas
rice_emission_factors	RICE emission factors
rice_turbines	RICE turbines
separators_heaters	Separators and heaters
tanks	Tanks
truck_loading	Truck loading
well_completions	Well completions

2. Read exported files into R.

   This step is handled automatically by R's built in `read.csv` function, assuming that the tables have the names listed in Table 3.2.

3. Determine the number of wells associated with each unique facility ID and calculate the well fraction.

   A conversion must be made to switch from the "facility" basis used in the OGEI database to the well-by-well basis used throughout the model. This is accomplished by counting how many unique API numbers are associated with each facility and operator ID key value combination in the `apis` table (facility and operator ID are combined to make a single key, referred to hereafter as the facility-operator key). Counts are made using an internal function called `cptable`, which uses a SQL query to (a) find each unique combination of values over a specified set of columns and (b) counts the number of repetitions for each combination. That count is inverted (i.e. calculating the 1/count value for each key) to find the well fraction. For example, if a particular key value has four unique API numbers associated with it, then the well fraction would be $1/4 = 0.25$. Later on in the MC simulation, if a well selected that key value from a CPT, then the emissions would calculated for that entire facility, but only 25% of those emissions would be assigned to the individual well in question. Well fraction is included in every CPT under the column "wfrac" by using the internal function `apimerge` to match wfrac values by facility-operator key.

4. Match key values in the `production_areas` table to the facility-operator key used in all other tables.

   The `production_areas` table uses operator ID and production area as keys. In order to match the VOC wt% and natural gas molecular weight in that table with the default facility-operator key used in all other tables, we need to perform a merge. The `facilities_list` table contains the information necessary to make the connection. A table called "prodlist" is created by merging these two tables together, returning a table with columns for operator, production area, the facility-operator key, gas molecular weight, and finally VOC wt%. This table is referenced as part of the CPT data analysis for pneumatic pumps and fugitives equipment types.

5. Make the well completions CPT.

   Using the `cptable` function, a CPT is made from the `well_completions` table using the columns:

   - annual_diesel_usage
   - flare_efficiency
   - pct_control

   Well fraction isn't considered as a factor for well completion because each reported well completion is assumed to be for a single well (i.e. wfrac = 1). Additionally, any record in the `well_completions` table which is reported to have zero annual diesel usage is excluded from the CPT (on the assumption that every well completion requires non-zero fuel consumption).

6. Make the RICE and Turbines CPT.

   Two tables contain the required information about the RICE and turbines equipment type: `rice_turbines` and `rice_emission_factors`. The tables are connected via the "rice_id" key. The first step therefore is to merge these two tables together. However given the structure of the `rice_emission_factors` table, the merge must be executed for each pollutant species (PM_{10}, $PM_{2.5}$, SO_x, NO_x, etc.). After that series of merges is completed, wfrac is added via the `apimerge` function, the rice_id column is dropped, and a CPT is made from the merged table using the `cptable` function on the columns:

   - horsepower

- annual_hours
- total_combusted
- fuel_heating
- control_type
- control_pm10
- control_pm25
- control_sox
- control_nox
- control_voc
- control_co
- control_ch2o
- wfrac
- fpm10
- upm10
- fpm25
- upm25
- fsox
- usox
- fnox
- unox
- fvoc
- uvoc
- fco
- uco
- fch2o
- uch2o

7. Make the Separators and Heaters CPT.

   Using the `apimerge` and `cptable` function, a CPT is made from the `separators_heaters` table using the columns:

   - heat_duty
   - hours_operation
   - fuel_heat
   - control_status
   - percent_control

- wfrac

8. Make the Dehydrators CPT.

   Using the `apimerge` and `cptable` function, a CPT is made from the `separators_heaters` table using the columns:

   - hours_operation
   - control_type
   - percent_control
   - heat_input_rate
   - pilot_volume
   - factor_voc
   - wfrac

9. Make the Tanks CPT.

   Using the `apimerge` and `cptable` function, a CPT is made from the `tanks` table using the columns:

   - control_type
   - control_percent
   - combustor_heat_input
   - pilot_volume
   - oil
   - total_voc
   - ratio
   - wfrac

   The "oil" and "ratio" columns are generated in the data analysis script. The "oil" column is the sum of the throughput_condensate and throughput_oil columns of the `tanks` table, and "ratio" is the ratio of the total VOCs (as reported in `tanks` total_voc column) divided by the "oil" column. Since some rows in the `tanks` table have 0 values for "oil" it's possible to have "ratio" values of ∞. To correct for this, any row in `tanks` which has either an "oil" value of 0 or "ratio" value which is NA are removed prior to calculating the CPT for tanks.

10. Make the Truck Loading CPT.

    Emissions from truck loading are accounted for based on the individual oil production rates from each well, therefore wfrac and the `apimerge` function aren't required and the CPT for truck loading is generated using `cptable` and the following columns from the `truck_loading` table:

    - s_factor
    - vapor_pressure
    - molecular_weight

- temp_r
- control_type
- control_percent

11. Make the Pneumatic Controllers CPT.

    Using the `apimerge` and `cptable` function, a CPT is made from the `pneumatic_controllers` table using the columns:

    - high_bleed
    - intermittent_bleed
    - low_bleed
    - avg_hours
    - wfrac

12. Make the Pneumatic Pumps CPT.

    In addition to using `apimerge` to add wfrac to the `pneumatic_pumps` table, we also require the gas molecular weight and VOC wt% data from the previously created prodlist data.frame. After performing both merges, the `cptable` function is used to create a CPT is made from the following columns:

    - annual_operation
    - vent_rate
    - control_type
    - control_percent
    - gas_mw
    - voc_wt
    - wfrac

13. Make the Fugitives CPT.

    Creating a CPT table for the `fugitives` table requires several steps. First, wfrac is added in using `apimerge`. Second, the component counts must be added from the `component_count` table, which uses the operator and production_designation column instead of the facility-operator key. The previously defined prodlist data.frame is used to match the two different key types. Finally, component counts are given for a variety of fluids (gas, heavy oil, light oil, and water/oil). Therefore the counts in the `component_count` table must be merged to the `fugitives` table once for each type of fluid, generating four different tables. `cptable` is then run on each table using the following columns:

    - production_hours
    - voc_wt
    - valves
    - pumpseals
    - other1

- connectors
- flanges
- open_lines
- wfrac

14. Save the results.

    All of the CPTs generated in the `emissionUpdate` function are structured into a list object called "eci" and are saved to disk.

Chapter 4

Monte-Carlo Simulation

The MC simulation takes the CDFs and regression results from the 2.x Data Analysis section of the model and then randomly and repeatedly calculates the economic and environmental impacts of oil and gas development. The random component in the simulation comes from the selection of values for each term from CDFs, such as the error in EIA price forecasts, the characteristics of each well (location, type, depth, etc.), and the production rates from each well (as calculated by decline curve coefficients). Based on the random and independent selection of the value for each term in each iteration or "run" of the MC simulation, a number of other terms are calculated based on the formulas fitted through regression in Section 2.x of the model, such as the drilling schedule, oil and gas production rates from existing wells, and the cost of operating each well. Each step in the MC simulation is discussed in detail below.

4.1 Transition to MC Simulation

This short code block at the start of the MC simulation transitions the `main` script from the data analysis section to the MC simulation section. A message is printed to the console noting the total elapsed time for completing the data analysis segment. Next, the start time of the MC simulation is noted in another message that is printed to the console and a progress bar is shown indicating the number of runs completed (as a percentage of the total number of runs specified for the MC simulation in `IO_options`).

4.2 Energy Price Path Simulation

The first step of the MC simulation process is to generate the energy price paths for both oil and gas for each run of the simulation. There are a number of possible options that can be used for generating the energy price paths, which are discussed below. Note that the forecasting method used for oil and gas prices are independent and are specified separately (should the user wish to use a different method for each).

1. Simulate using ARIMA model

   If this method is selected, the model simulates the energy price paths for oil and gas either the (a) automatic or (b) manually specified ARIMA model fit by the `ARIMAfitUpdate` function. This method should be used if performing a long-term forecast (5 - 20 years).

2. Simulate from EIA Annual Energy Outlook forecasts

   If this method is selected, the model simulates the energy price paths for oil and gas by adjusting the EIA AEO oil and gas forecasts by a randomly selected relative error percentage from the specified error CDFs. This method is recommended for making forecasts on the order of 1 - 5 years into the future.

   Two types of error matrices can be specified:

   (a) (Not recommended) Direct relative error, as given by Equation 3.7 (see Section 3.8).

   If this method is specified `Eoil` and `Egas` are used for the error CDFs. For each run, a single row is selected from each matrix (rows = CDF quantiles, columns = time steps, and values in the matrix = relative error values). The two row selections are made independently. The randomly selected rows are taken as the values for RE at each time step in the simulation, and the EIA forecast is adjusted as specified by Equation 3.10 (i.e. $SP = FP * (1 + RE)$).

   (b) (Recommended) Fractional relative error, as given by Equations 3.8 and 3.9 (see Section 3.9).

   Like the direct relative error method, a single row is selected independently from each of the relative error matrices `EoilFrac` and `EgasFrac`, representing the values of RE as a function of time at a randomly selected CDF quantile. The selected RE values adjust the EIA forecast by either Equation 3.11 ($SP = FP/RE$) or Equation 3.12 ($SP = RE * FP$). Since RE is always a value between 0 and 1, Equation 3.11 will push the forecasted price upwards, while Equation 3.12 will reduce the forecasted price.

   The choice of which equation to use is determined randomly by picking a value from a binomial distribution, which returns either a value of 0 or 1. The probability that a value of 1 is returned is specified by the user in `IO_options` (by default, the probability is 50%). When a value of 1 occurs, the model uses Equation 3.11 (adjusting the forecast upwards). Conversely, a selection of the value 0 causes the model to use Equation 3.12 (adjusting the forecast downwards).

3. Use the actual energy price path

   If this option is selected then the energy price paths for oil and gas will be the actual energy price path for oil and gas in every run of the simulation. This method can only be selected if the model is being run in cross-validation mode, and is intended for testing the accuracy of the drilling schedule model.

4. Constant prices equal to the average of the last n months.

   This option is intended for forecasts that are $\leq$ 1 year. Over a short-term period, it's reasonable to assume that the future price will be constant and equal to the average of the last n months. The value of n can be specified by the user. The recommended value is 12 months.

5. Use a user-specified price path

   If this option is selected, the model will use whatever price path is specified in `IO_options`. This option is intended to allow the user to analyze "what-if" scenarios, such as modeling what would happen if

oil prices were $50/bbl for the entire simulation period. The specified price paths for oil and gas must be given in the form of a matrix, with one row for each MC iteration and one column for each time step, with the value of oil or gas prices in each element of the matrix in units of $/bbl or $/MCF, respectively.

A wrapper function called `EPFsim` creates the simulated price paths. The input options for this function are:

1. The energy price path simulation method (`opt$epf.type`), selected from one of the methods described above.
2. Whether to use the direct or fractional relative error model (`opt$epf.EIA.type`), if the EIA forecast method is selected.
3. The binomial probability for the fractional relative error model (`opt$epf.EIA.fracProb`), if the EIA forecast method is selected.
4. The number of months n to take the average price over (`opt$epf.const`), if the constant price method is selected.
5. The user price paths for oil and gas (`opt$epf.uepf`), if the user-specified method is selected.

4.3 Well Drilling Schedule

The drilling schedule simulation function `drillsim` calculates the number of wells drilled in response to the simulated energy price paths for oil and gas using one of the fitted models from `drillingModelUpdate` (Equations (3.3)-(3.6)). Two alternative options exist. In the first, the user may specify that the actual drilling schedule be used. This option can only be selected if the model is being run in cross-validation mode, and is intended for testing the accuracy of predicting oil and gas production from new wells (by eliminating all of the variability in the drilling schedule). The second alternative is that the user may directly specify the total number of wells drilled each month. This option is intended to serve the same "what-if" purpose as the equilvalent option for energy prices.

The input options for this function are:

1. Which drilling simulation method to use (`opt$DStype`). Valid options are simulation, actual well counts (in cross-validation mode only), and user-specified.
2. The distributed-lag model to use for calculating the drilling schedule (`opt$DSsimtype`). Any of the models discussed in Section 3.5 can be used.
3. The user-specified drilling schedule (`opt$DS.uspec`), for use with the option of the same name.

4.4 Prior Wells

There are several calculation steps for handling information about prior wells. Unlike new wells, the characteristics and production from prior wells need only be calculated once (instead of being randomly generated and calculated during each run of the MC simulation). The results of the calculations from prior wells can then be referenced during each run of the MC simulation (such as when calculating total production). The functions used to calculate production and collect information about prior wells are described below.

4.4.1 `priorProd`

This function calculates the oil and gas production for each prior well in the Uinta Basin by extrapolating from each well's last hyperbolic decline curve fit (Equation (3.13)). The DCA results data.frames `mo` and `mg` contain all of the necessary fitted decline curve coefficients, as well as the age of each well at the start of the simulation period, so the extrapolation is fairly straightforward. The only issue with production from prior wells is that approximately 18% of prior wells don't have a fitted last hyperbolic decline curve. Almost all of these wells are "skipped" during the DCA because they have fewer non-zero production points than are required by the DCA algorithm. Production from skipped wells is estimated using the same method applied to new wells (simulated decline curve coefficients generated randomly during each run for use with Equation (3.14)). The `priorProd` function begins this calculation process for prior wells by identifying each prior well that is skipped and collecting basic information about those wells.

The input options for this function are:

1. How old a well without a fit can be and still be included in the population of prior wells (`opt$tend.cut`).
2. How far into the future production should be calculated for each prior well (`opt$MC.tsteps`).

4.4.2 `priorInfo`

This function collects details about each prior well that are needed for calculating economic and environmental impacts in the MC simulation. Specifically, it collects the well API number, field number, well type, well depth, and the surface lease owner for each prior well.

No input options are required for this function.

4.5 Monte-Carlo Loop

In the MC loop, detailed calculations are carried out on a well-by-well basis for each time step in the simulation period. Well characteristics are randomly generated from the appropriate CDFs, production calculations are made for new wells (and for prior wells without fits), and the subsequent economic and environmental impacts are calculated.

Prior to starting the MC loop, the matrices and lists used to save the results of each run are preallocated to the workspace (to speed up the MC simulation by preventing the resizing of result objects at the end of each loop iteration). The result objects are:

1. `osim` - a matrix of oil production volumes (in units of barrels) from all new wells. The dimensions of `osim` are set by the input options `opt$nrun` (for rows) and `opt$MC.tsteps` (for columns).
2. `gsim` - a matrix of gas production volumes (in units of MCF) from all new wells (same dimensions as `osim`).
3. `posim` and `pgsim` - matrices for production volumes of oil/gas from all prior wells with the same units and dimensions as `osim` and `gsim`.
4. `aE` - a structured list of activity-based emission results, containing total and reduced emissions for each pollutant species, the fraction of emissions from new wells and prior wells, and the fraction of emissions from each source activity. Most components of the `aE` list are matrices with the same dimensions as `osim`.
5. `eE` - a structured list of equipment-based emission results, containing total emissions by species and by equipment type. All components of the `eE` list are matrices with the same dimensions as `osim`.
6. `reE` - copy of eE used for storing equipment-based emission results with reductions applied.

The details of each step of the MC loop are described below.

4.5.1 Well Data Simulation

This section of the MC loop generates the data.frame `wsim`, which contains all of the randomly generated details about each new well in each MC run. Each step in the creation of `wsim` is described below (note - except where stated otherwise, the functions involved in creating `wsim` do not require any input options).

1. Create `wsim` and column `tDrill`.

   The first step is to create the data.frame `wsim` and define the time step at which each well is drilled. This is done using the function `sim_tDrill`, which essentially generates a data.frame with one row for each well in the drilling schedule for the current MC run and one column specifying the time step in which it was drilled. For example, if the drilling schedule for a MC run called for two wells to be drilled in time step 1, three wells in time step 2, and one well in time step 3, then the `tDrill` column in `wsim` would look like: [1 1 1 2 2 3].

2. Pick the field number for each new well.

   The first item that is randomly generated for each well in `wsim` is field number (i.e. location). This term is selected from the CDF contained in the data.frame `cdf.ff`. For each well, the function `sim_fieldnum` generates a random number between 0 and 1 and compares that to the cumulative probabilities in `cdf.ff` and picks the field associated with that random number.

3. Pick the well type for each new well.

   Once the field number is selected, the model can then randomly pick the well's type (oil, gas, or dry) using the `sim_wellType` function. A random number is picked for each well and compared to the cumulative probabilities for well types by field number contained in the data.frame `prob`.

4. Pick the time delays for oil and gas production from new wells.

   Some wells have a time delay between when they are first drilled and when they begin producing. The CDFs for time delays by well type and field are contained in the list objects `DCA.cdf.coef.oil` and `DCA.cdf.coef.gas`. The function `sim_tdelay` randomly picks the time delays for each well for both oil and gas production. Note that the time delays for each type of production are separate and independent. For example, a well could have a time delay of 0 for oil production (i.e. no time delay) but 3 for gas (i.e. three months). Production curves for wells with time delays are started at time step $t_{Drill} + t_{delay}$.

5. Pick well depth.

   Well depths are randomly selected from CDFs based on well type (oil or gas) by the function `sim_depth` (well depths were found to vary more as a function of well type rather than field number).

6. Pick the surface lease owner.

   The function `sim_lease` picks the surface lease owner (federal, state, Indian, or fee) based on each well's field number as specified by the CDFs in the data.frame `cdf.flt`.

7. Pick rework times for new wells, existing wells with fits, and existing wells without fits.

   Any well in the simulation (new or prior) could potentially be reworked. The function `sim_rework` randomly picks if and when a rework occurs from the CDF `cdf.rework` based on well type, returning the age of the well at which the rework occurs. It's possible that the randomly selected rework time occurs before or after the simulation period, in which case the rework is effectively ignored. For example, suppose that an existing well were 100 months old at the start of a 60 month simulation period. Any rework time picked for that well that is less than 100 months or greater than 160 months is outside of the time period simulated and is therefore ignored. Rework times are picked separately for new wells, prior wells with fits, and prior wells without fits because each contains slightly different formats for storing information about the well (field number, well type, depth, etc.).

   Additionally, since this is the first instance in which a randomly selected property is being generated for prior wells, a new data.frame called `wpri` (well prior information) is created to hold the rework times (and other subsequent information) for prior wells.

   The `sim_rework` function requires the number of simulated time steps (`opt$MC.tsteps`) and the simulation start date (`opt$tstart` - only necessary for prior wells).

8. Combine well information about skipped wells with `wsim`.

   Since prior wells without fits have their production rates simulated in the same way as new wells, this step adds all skipped wells as additional rows to the `wsim` data.frame.

9. Duplicate reworked wells.

   Reworked wells (from all well types) are duplicated as additional rows and added onto the bottom of `wsim` by the `sim_dupRework` function.

10. Pick decline curve coefficients.

    The `sim_DCC` function picks the decline curve coefficients used to estimate production for every well in `wsim`. The function allows for the use of either hyperbolic (Equation (3.13)) or cumulative (Equation (3.14)) types of decline curve coefficients. Details of the decline curve selection methods are covered in Chapter 5.

    The input options for this function are (a) the type of decline curve coefficient to pick for both oil (`opt$mcDCCpick.type.oil`) and gas (`opt$mc.DCCpick.type.gas`) production, and (b) the sequence of time steps used in the simulation (`opt$tsteps`).

11. Pick activity-based emission factors for all wells.

    Emission factors for emissions in terms of carbon dioxide equivalence (CO_2E), methane (CH_4) and non-methane volatile organic compounds (VOCs) are calculated for each of the following activity categories (based on the emission factors reported in Wilkey et al. [2016]):

    - Site preparation
    - Transportation of materials for...
      - Drilling
      - Completion
      - Rework
      - Production
    - Completion
    - Gas production
    - Gas processing
    - Gas transmission and distribution
    - Oil production
    - Oil transport by tanker truck

    The mean and standard deviation for each factor and each pollutant are specified in the data.frame `opt$EF`, and are used by the function `sim_EF` to randomly pick emission factors for each well, emission type, and activity category (assuming a normal distribution). To simplify the number of emissions factors used, several activities are summed together to give one single emission factor. Specifically, the emission factors for site preparation and transportation of materials for drilling, completion, and production are all summed together into a single emission factor called `EFdrill`. Likewise the emission factor `EFrework` includes the transportation of materials for both completion and reworking.

12. Pick equipment-based emission calculation inputs

    All of the inputs for calculating equipment-based emissions are selected by the function `sim_eq_EF`, which randomly selects one row from each CPT in the list `eci` for each well. The randomly selected rows are added to `wsim` and `wpri`, and are later referenced when calculating equipment-based emissions.

    No input options are required for this function.

4.5.2 Production Simulation

Two functions are used to calculate production volumes:

1. Production from new wells is calculated by the function `productionsim`. The function can calculate production using either the hyperbolic or cumulative production equations (specified by the input option `opt$mc.DCeq.type`) by using the corresponding simulated decline curve coefficients contained in the `wsim` dataframe to project production over a specified number of time steps (as given by `opt$MC.tsteps`).
2. While the production from existing wells is calculated before the start of the MC loop by `priorProd` (see Section 4.4.1), there is still the potential for variation in production from individual existing wells due to reworks. Any existing well which is randomly selected for reworking in a given iteration of the MC loop has its production zeroed out beginning in the month that the well is reworked using the function `priorProdReworkAdjust`. The only required input option for this function is the number of time steps in the simulation (`opt$MC.tsteps`).

4.5.3 Lease Operating Costs

The next step in the MC loop is to calculate the lease operating costs for each well (`LOC`). These costs are estimated using the regression model fitted to the lease operating cost data from U.S. Energy Information Administration [2010] as discussed in Section 3.4. The function `LOCcalc` calculates lease operating costs for all wells using the fit results the `leaseCostUpdate` function and each MC loop's randomly selected inputs for Equations 3.1 and 3.2 (i.e. well depth, oil/gas price, and oil/gas production volumes).

No input options are required for this function.

4.5.4 Production Correction

The production rates calculated from the `productionsim` and `priorProd` function assume that a well produces indefinitely, even if it is producing at a rate that is so low that it is no longer economical. To correct for wells that are producing at uneconomical rates, the next step in the MC loop is to calculate the gross revenue (`gr`) from each well and compare that to each well's lease operating costs (`LOC`). Any well with a ratio of `LOC`/`gr` that is greater than a specified cutoff value (set by the user in `IO_options`) is assumed to be permanently shut-in and abandoned (i.e. no further production or costs). In the code this is accomplished by finding all of the records above the specified threshold and setting the production and lease operating costs of those wells to zero.

4.5.5 Activity-Based Emissions

Activity-based emissions from each well are calculated using the `Ecalc` function. Emissions for each type of pollutant and each activity category are tracked in matrices, with one row for each well and a column for

each time step. Emissions fall into two categories:

1. One-time events, including drilling, completion, and reworks. Emissions in this category are handled by simply placing each one-time emission factor that was randomly generated in `wsim` or `wpri` into the emissions matrix for each well (row) in the time step (column) associated with that event.
2. Production events, including production, processing, etc. Production events for gas have a flat emissions rate that isn't dependent on gas production (the emission factor is in units of metric tons/well/year). Therefore the calculation step for gas production emissions checks if each well is producing gas, and if it is, then the randomly generated emission factor for that well is placed in the emissions matrix (for each time step as appropriate). Gas processing and transmission and distribution steps are dependent on gas production rates, and so their calculation is handled by multiplying the emissions factor by gas production rates for each well. Emissions from oil production for transportation are similarly dependent on oil production rates and are calculated in the same manner.

After calculating the base emissions rates for each pollutant for each activity category, the `Ecalc` function then sums together all of the emissions matrices by pollutant type to give the total emissions for each pollutant species by well and time step.

The last step in `Ecalc` is to calculate the potential reductions from new source performance standards (NSPS). Each NSPS has to be programmed manually into the `Ecalc` program, but the basic algorithm used is to:

1. Calculated the time step in the simulation at which the emission reduction well occur. For example, one of the emissions reductions currently programmed into `Ecalc` is an NSPS reducing emissions from the production, processing, and transmission of gas. This NSPS went into effect for wells drilled on or after Nov. 2012. If the model were simulating the time period Jan. 2010 to Dec. 2014, then the time step (in the simulation) where the NSPS would take effect is month 23.
2. Find all the wells which are effected by the NSPS. Continuing the example of the NSPS from Nov. 2012, this would be all wells drilled or reworked on or after month 23 of the simulation.
3. Multiply the base emissions matrices (by species and activity) by the NSPS emissions reduction factor (as specified in `IO_options`). A function called `redfun`, defined internally in `Ecalc`, can be used to apply the reduction factor to only those time steps on or after the NSPS implementation date.
4. After applying all desired emission reductions, sum together the emission matrices to produce a single set of emission matrices by species.

4.5.6 Equipment-based Emissions

Equipment-based emissions (EBE) are calculated for each well using the function `eqEcalc` using (a) the randomly selected rows from each CPT (contained in the `wsim` and `wpri` dataframes), (b) the production volumes of oil and gas, and (c) all of the options in `EF_options` related to emissions calculations. EBEs

are calculated for each type of equipment following the same procedure used in UDAQ's OGEI spreadsheet. Each step in the EBE calculation process is outlined below.

1. Preliminary steps (used by most of the subsequent EBE calculation steps)

   (a) Create a production identity matrix (`Eid`)

   The first step in the calculation procedure is to create a matrix with one row for each well and one column for each time step where the value in each element is 0 if a well isn't producing either oil or gas and 1 otherwise. Since none of the EBEs depend on oil or gas production volumes (except VOC emissions from tanks), EBEs are either "on" or "off." Most of the functions used to calculated EBEs for each type of equipment simply return a single annual emissions amount. This is translated into a monthly emissions total for each well by multiplying annual emission by the `Eid` matrix (and converting to a monthly emissions rate).

   (b) Define the reduced emissions function `redfun`

   This function selects a subset of rows from a matrix (given by the input `ind`) and multiplies them by $(1 - r)$ where r is the emissions reduction (specified as a fraction) specified by the user in `EF_options`. Additionally, the function is designed so that the reduction can apply to a subset of the columns of the original emissions matrix (i.e. the reduction can be applied after a certain date in the MC simulation).

   (c) Define the reduction index selection function `ind.select`

   The `redfun` applies reductions to all of the rows specified by the row index input `ind`. This function handles three possible selection categories used to create the `ind` vector: well type (oil or gas), jurisdiction (federal, state, fee, or Indian), and county (Uintah or Duchesne). Any combination of these categories can be given to `ind.select` (including the options of selecting all of the above) and the function will return the matching row indices.

   (d) Get the maximum production rates of oil and gas for each well

   Several EBE functions required the maximum potential production rates of oil and gas for each well. This step finds those max rates and defines them as `moil` and `mgas`.

2. Calculate emissions from well completions (i.e. new wells and reworks)

   Well completions are one-time emission events. Whenever a new well is drilled or an existing well is reworked, the emissions from that event are calculated using the equation:

$$E_i = F * EF_i * \left(1 - \frac{r}{100}\right) * \frac{1}{2000} \tag{4.1}$$

   where E_i is the emissions of species i (in tons), F is the diesel fuel usage (in gallons), r is the emissions reductions (due to the use of control equipment, if any), and EF_i is the emission factor (in lb/gallon). In the OGEI database, r is reported as a percentage. Therefore in the calculated r must be divided by 100 to convert to a fraction. The pollutant species for well completions are: PM_{10}, $PM_{2.5}$, NO_x, VOCs, and CO.

   In the code, an identity matrix (similar to `Eid`) is created called `Ewc` which contains one row for each well, one column for each time step, and which contains zeros for all elements except the for the time

step in which each well is drilled, which is instead recorded as a 1. The EBEs are calculated for each well using Equation 4.1. Those results are then multiplied by `Ewc`, and the resulting matrices are saved as the base emissions results list `E`.

Note: the process described below for calculating reduced emissions applies to all types of equipment.

Any applicable reduction selection criteria specified in `EF_options` for well completions are then used to find which rows to apply reductions to using `which()` and `ind.select`. The reduction selection criteria include:

(a) The time step in which a well is drilled

(b) The start/implementation date of the reduction

(c) Well type (oil, gas, or both)

(d) Jurisdiction (federal, state, fee, Indian, or all)

(e) County (Uintah, Duchesne, or both)

(f) Minimum annual emission thresholds (e.g. wells with emissions greater than some specified amount)

(g) Minimum monthly oil or gas production rates

Additionally, any well which selects a row of inputs from the CPT which already has some form of emissions control (combustor, VRU, etc.) will be automatically excluded from having any further emission reductions applied to it (to prevent double-counting reductions).

Based on the selection criteria, reduced emissions are calculated for any applicable rows using `redfun` and the reduced emissions are saved in the list `rE`.

3. Calculate emissions from RICE and Turbines

RICE (reciprocating internal combustion engines) and turbines have ongoing emissions which are calculated by the function `calc_E_rt` using either equation:

$$E_i = F * HV * EF_i * (1 - r) * \frac{w_{frac}}{2000} \tag{4.2}$$

if the emission factor EF_i is in units of lb/MMBtu, or:

$$E_i = \left(\frac{hp * t * EF_i}{453.592}\right) * (1 - r) * \frac{w_{frac}}{2000} \tag{4.3}$$

if the emission factor EF_i is in units of gal/hp-hr. In both of the above equations F is the total fuel combusted (MMCF/yr), HV is heating value (Btu/CF), hp is horsepower, t is operation time (in hours/year), r is the control reduction (expressed as a fraction), and w_{frac} is the "well fraction" discussed in Section 3.14. The pollutant species for RICE and turbines are: PM_{10}, $PM_{2.5}$, SO_x, NO_x, VOCs, CO, and CH_2O.

Prior to using `calc_E_rt`, any NA values in the randomly selected RICE and turbine CPT row selections for each well are replaced with zeros using `NA.overwrite`. The cleaned up inputs are run through `calc_E_rt`, returning annual emission rates for each pollutant species for each well. These vectors are

converted to a monthly basis and multiplied by `Eid` to create the base emissions matrices for RICE and turbines, which are added to `E`.

Finally, the same emissions reductions process described for well completions is applied to the base emissions to find `rE`.

4. Calculate emissions from separators and heaters

   Separators and heaters have ongoing emissions which are calculated with the function `calc_E_sh` using the equation:

$$E_i = \frac{HD * t * EF_i}{HV} * \frac{w_{frac}}{2000} \tag{4.4}$$

   where HD is the heat duty rating (MMBtu/hr) and all other terms are identical to those previously identified. The pollutant species for separators and heaters are: PM_{10}, $PM_{2.5}$, SO_x, NO_x, VOCs, and CO. Note that separators and heaters in the OGEI database include the option of using low NO_x burners, but no other built-in control equipment. Therefore for NO_x, the emissions are calculated as:

$$E_i = \frac{HD * t * EF_i}{HV} * \frac{w_{frac}}{2000} * (1 - r) \tag{4.5}$$

   where r is the control reduction (expressed as a fraction).

   The steps for calculating emissions for separators and heaters is essentially identical to those for RICE and turbines. NA values from the randomly selected rows of the CPT are replaced with zeros, `calc_E_sh` is used to find annual emissions for each well and species, the results are multiplied by `Eid` and converted to a monthly basis to find the base emissions `E`, and finally the reduced emissions for wells which meet selection criteria is calculated using `redfun`.

5. Calculate emissions from dehydrators

   Dehydrators have ongoing emissions which are calculated with the function `calc_E_dh` using either the equation:

$$E_i = \left(\frac{HD * 8760 * EF_i}{HV} + \frac{V_{pilot} * EF_{i,pilot}}{10^6} \right) * \frac{w_{frac}}{2000} \tag{4.6}$$

   for NO_x and CO, or:

$$E_{VOC} = \left(EF_{VOC,DH} * t + \frac{V_{pilot} * EF_{VOC,pilot}}{10^6} \right) * \frac{w_{frac}}{2000} \tag{4.7}$$

   for VOCs. In both of the above equations, V_{pilot} is the volume of gas used by the combustor pilot (CF/yr), and the subscripts DH and $pilot$ indicate the emissions factor is for the dehydrator or combustor pilot, respectively. All other terms are identical to those defined previously. The first term in Equation 4.6 is for the emissions related to the operation of the combustor, and the second the combustor pilot.

   Like separators and heaters, the steps for calculating emissions from dehydrators are identical to that used for RICE and turbines.

6. Calculate emissions from tanks

   Tanks have ongoing emissions which are calculated with the function `calc_E_tank` using a combination of (a) the combustor and combustor pilot emissions equations (Equations 4.6 and 4.7), and (b) the VOC emissions ratio discussed in Section 3.14 (e.g. $R = (VOC)/(oil)$). In the `calc_E_tank` function, the combustor and combustor pilot emissions are first calculated (and converted to a monthly basis inside the function). Next, the VOC emissions from the tank are calculated using the equation:

$$E_{VOC,tank} = R * [\texttt{osim}] * (1 - r) \tag{4.8}$$

   where R is the VOC emissions ratio drawn from the CPT, `osim` is the oil production matrix for all wells (rows = wells, columns = time steps, elements = monthly oil production volumes), and r is the control reduction (expressed as a fraction). Any element in $E_{VOC,tank}$ which is greater than the maximum reported monthly emissions rate in the OGEI database is replaced with that maximum rate. Tank VOC emissions from the combustor and pilot calculated in Equation 4.7 are then added to $E_{VOC,tank}$ to get total tank VOC emissions. The remainder of the calculation process for EBEs from tanks is identical to those described for RICE and turbines (with the exception that conversion from an annual to a monthly basis isn't required).

   The pollutant species for tanks are: NO_x, VOCs, and CO.

7. Calculate emissions from truck loading

   Truck loading produces ongoing emissions which are calculated with the function `calc_E_truck` using the equation:

$$EF_i = \left(\frac{12.46 * S * P_{vap} * MW}{T}\right) * (1 - r) * \left(\frac{42}{10^3 * 2000}\right) \tag{4.9}$$

   where EF_i is the emission factor for truck loading (ton VOC/bbl) S is the "S" factor for the truck loading rack mode of operation, P_{vap} is vapor pressure (psia), MW is molecular weight (lb/lbmol), T is temperature (°R), and r is the control reduction (expressed as a fraction).

   The only pollutant species considered from truck loading are VOCs.

   After calculating the value of EF_i for each well, the oil production matrix `osim` is multiplied by the EF_i vector to find VOC emissions as a result of truck loading. The remainder of the calculation process for EBEs from truck loading is identical to those described for RICE and turbines.

8. Calculate emissions from pneumatic controllers

   Pneumatic controllers have ongoing emissions which are calculated with the function `calc_E_pctrl` using the equation:

$$E_{VOC} = (n_H * EF_H + n_I * EF_I + n_L * EF_L) * t * w_{frac} \tag{4.10}$$

   where n is the number of high (H), intermittent (I), or low (L) bleed devices with a VOC emissions factor EF for each type. All other terms are identical to those previously defined.

   The only pollutant species considered from pneumatic controllers are VOCs.

After finding the total annual VOC emissions from pneumatic controllers as calculated by `calc_E_pctrl`, the remainder of the calculation process is identical to that described for RICE and turbines.

9. Calculate emissions from pneumatic pumps

   Pneumatic pumps have ongoing emissions which are calculated with the function `calc_E_ppump` using the equation:

$$E_{VOC} = \frac{VR * 60 * MW * t * m_{VOC}}{379 * 2000} * w_{wfrac} \tag{4.11}$$

   where VR is the average vent rate for pumps (CF/min), m_{VOC} is the VOC weight fraction, and all other terms are identical to those previously described.

   The only pollutant species considered from pneumatic pumps are VOCs.

   After finding the total annual VOC emissions from pneumatic pumps as calculated by `calc_E_ppump`, the remainder of the calculation process is identical to that described for RICE and turbines.

10. Calculate fugitive emissions

    Fugitive sources produce ongoing emissions which are calculated with the function `calc_E_fug` using the equation:

$$E_{VOC} = \frac{EF_{fug} * t}{2000} * w_{frac} \tag{4.12}$$

    where EF_{fug} is the fugitive emission factor (and all other terms are identical to those previously described):

$$EF_{fug} = \sum_{i=1}^{k} n_i * EF_i \tag{4.13}$$

    where n is the number and EF the emission factor of each i type of fugitive source. Fugitives sources considered here are: valves, pump seals, connectors, flanges, open lines, and a catch-all "other" category.

    The only pollutant species considered from fugitive sources are VOCs.

    Fugitive source counts in the OGEI database are recorded based on the fluid type (gas, heavy oil, light oil, and water/oil) for each production area. Consequently `calc_E_fug` must be called for each fluid type and the results summed together to get the total annual fugitive emissions for each well. That vector is then multiplied by `Eid`, converted to a monthly basis, etc. following the same procedure outlined for RICE and turbines.

11. Calculate total emission results

    At the conclusion of the EBE calculation process, the function `eqEcalc` has created two structured lists, `E` and `rE` (for base and reduced emissions, respectively), with one sub-list for each type of equipment containing one matrix for each type of pollutant generated by that equipment. These matrices have one row for each well in the MC simulation and one column for each time step.

    The final step in the function is to sum the emissions together from all wells into (a) total emissions by species and (b) total emissions by species and equipment type. This summation is handled using the

function `etotal`, which takes the column-sums of the well-by-well emission results in `E` and `rE`. These summation results are then returned to `main`.

4.5.7 Totals

The last step in the MC loop is to sum together the detailed results for each run into a single total for the entire Uinta Basin as a function of time (e.g. total oil and gas production, emissions of each type of pollutant species, etc.). The results that are summed together this all of the objects predefined at the start of the MC loop (e.g. `osim`, `gsim`, `aE`, etc.) as discussed at the beginning of Section 4.5.

4.6 Post-processing

At the conclusion of the MC loop, the script prints to the console the elapsed time for the MC loop and a message notifying that post-processing has begun. The script then sources the `postProcess` script (see Part V).

4.7 Save Results

Finally, at the end of the `main` script, the `if (opt$save == TRUE) {...}` statement saves the entire workspace to disk (if specified by the user in `IO_options`). The *.rda file saved to disk is the same file that can later be loaded at the end of the 1.x section of `main` as discussed in Section 2.4.

Part III

IO_options

Chapter 5

IO_options

IO_options is the second main script in the model, and contains all of the user input options that control the execution of functions in main (except for those specifically related to equipment-based emissions options, see Part IV). Options are organized into sections which (as closely as possible) follow the same structure as the code in main. All of the input options are in IO_options are described in detail below.

5.1 Global Options

The first section of IO_options sets "global" options for the simulation that effect all areas of the model or that are used by multiple functions in main. The options are:

- opt <- NULL

  The very first command in IO_options, this line creates the list object "opt" to which all subsequent options in IO_options are added. This line must be present and should not be changed.

- opt$save

  This option specifies whether or not the model workspace should saved at the conclusion of the main script. Valid options are TRUE or FALSE.

  opt$save.name

  Specifies the name of the file under which the model workspace will be saved if the previous option is set to TRUE.

  opt$load.prior and opt$load.name

  The counterpart of the save options above, this option indicates whether the results of a prior run should be loaded and if so what file should be loaded. opt$load.prior is a TRUE/FALSE input, while opt$load.name can be any character string under which a prior run results were saved.

- opt$file_ver

This input option controls a character string that is added onto the end of every *.rda file saved in the data analysis section of `main`. If a new analysis is being performed the value of this option should be changed. The value can also be changed to any previous version to load the results of a previous version of the data analysis functions.

- `opt$nrun`

  This options specifies how many MC simulation iterations or runs should be performed in the MC loop discussed in Section 4.5. Setting higher values for this option will lead to a more thorough sampling of the model's parameter space, but it will also linearly increase processing time. At the time this manual was written, each run through the MC loop required approximately 8 seconds on a mainstream desktop computer. For development and testing purposes setting this value to 100 is usually sufficient to achieve a rough sampling of the parameter space. Setting the value to 10^3 substantially smooths out the quantile results. It should also be noted that while there is little difference between the median results of the 10^2 vs. 10^3 simulation results, the extremes of the quantile distribution (the 10^{th} and 90^{th} percentiles) do spread farther out as the number of runs increases. The only hard upper limit to the value of `opt$nrun` is computer memory. Each iteration adds an additional row to all of the matrices containing MC loop simulation totals, consuming about 30 MB of memory per run.

- `opt$crossvalid`

  Logical TRUE/FALSE option that acts as a flag to control execution of queries and plotting options in `postProcess`. If set to true, the model will attempt to calculate the actual values of energy prices, oil and gas production, etc., over the simulation time period. If false (i.e. if the model is being run in predictive mode), all of these steps will be skipped.

- Time options

  These options control what time periods are used for the simulation period as well as the global training period (i.e. what time period is used in general for selecting the data used in the data analysis section of `main`). All dates must be entered in "YYYY-MM-DD" format. The specific options are:

  - `opt$tstart`

    The start of the simulation time period

  - `opt$tstop`

    The stop of the simulation time period

  - `opt$train.start`

    The start of the training period. The recommended value here is for "1984-01-01" (the first month for which data is available in the UDOGM databases.

  - `opt$train.stop`

    The stop of the training period. This should be set to a date which is both (a) one month prior to the simulation period (`opt$tstart`) and (b) for which data is available. For example, if the simulation period was going to start on "2010-01-01" then this value should be set to "2009-12-01."

  - `opt$tsteps`

    This option creates a vector of dates by month between `opt$tstart` and `opt$tstop`. It does not need to be changed.

- `opt$MC.tsteps`

  This option calculates how many time steps are in the simulation period. It does not need to be changed.

- Inflation adjustment

  All of the dollar values used in model calculations need to be on the same time basis. The two global options that control what time basis is used for inflation adjustment of dollar values are:

  - `opt$cpi`

    Enter the consumer price index (CPI) value you'd like to use as the time basis for dollar values here. CPI values are listed in Table 24 of the U.S. Bureau of Labor Statistics [2015] monthly CPI reports. Every function in the model that involves an inflation adjustment step will use this value.

  - `opt$cpiDate`

    Character string indicating what year or date corresponding to the value of `opt$cpi`. This value is only used in the plots produced by `postProcess` in y-axis labels (e.g. entering "2014" here would produce the y-axis label "2014 USD").

- `opt$xq`

  This vector specifies which cumulative probability points should be used to generate CDFs with the `CDFq` function. Specifying additional points will lead to finer resolution and selection of values for terms described by CDFs in the MC simulation. However if the number of cumulative probability points greatly exceeds the number of data points then the CDF will simply draw straight lines between each point (see Figure 2.2). Good results have been achieved with a probability step size of 10^{-4}. Lastly, the upper and lower limit of `opt$xq` should not be set to 0 or 1, as that would lead to values of $\pm\infty$ for CDFs generated from normal distributions.

- `opt$min.well.depth`

  This option controls subsetting options which can exclude wells that are shallower than the specified depth. Originally, this was used to skip wells that were only partially drilled. However subsequent investigation of the UDOGM database revealed that any well being held in confidential status has a reported well depth of 0 ft. Therefore to include information about these well's in the model this value should be left at 0.

- Flags for updating prepared data files

  This segment controls the execution of each data analysis function in `main`. Update flags are logical TRUE/FALSE values listed in `opt$[...].update` format, where "..." is the data analysis function that will be run if the flag is set to TRUE. If the flag is set to false the analysis function will be skipped and the results of the data analysis function saved under the version number specified in `opt$file_ver` will be loaded. In general, the best approach to updating the model is to (1) specify a new version number and then (2) set all update flags to TRUE. See Section 9.2 for detailed instructions on how to update the data and data analysis used by the model.

- Subsetting options for `production`

  This section lists all of the columns contained in the data.frame `production`, which contains the combined UDOGM database files `proddata`, `histdata`, and `welldata` for the entire state of Utah. When the subset `p` of wells located in the Uinta Basin is taken from `production`, only the columns in this options list which are uncommented will be included in `p`. There should be no need to change the selection of values in this segment unless the user wants to include new information from `production` in some future analysis.

5.2 Data Analysis Options

This section of `IO_options` controls the specific options for each data analysis function in `main`. The options for each function are described below.

5.2.1 `dogmDataUpdate`

`opt$psub` controls what subset to select from the data.frame `production`. Currently, the only valid subsetting option is "a," which corresponds to selecting only records from `production` that are located in the Uinta Basin (i.e. Uintah or Duchesne counties). Additional options can be added to the case-and-switch programming structure in `main`. If additional options are programmed into `main` in the future then this is the option to use to control which subset is selected.

5.2.2 `scheduleUpdate`

The options for this function are:

- `opt$SU.tsteps`

  This option controls what data is selected for use in `scheduleUpdate`. By default, it is set to use the same time range specified by `opt$train.tstart` and `opt$train.tstop`. Changing the value of this option isn't recommended.

- `opt$field.cutoff`

  As discussed in Section 3.2, not every field is worth analyzing individually. This option specifies the minimum value of the fraction (wells located in field i)/(total wells in Uinta Basin) that field i must have in order to be analyzed individually. The recommended value for this option is 0.05. Lower values can be selected, but the user may encounter errors with generating CDFs if too few values exist for terms in smaller fields. For example, suppose a field contained only 100 wells, 10 of which were oil wells, and that those oil wells didn't have enough production records to be fit in the `DCAupdate` function. No CDFs could be generated for oil decline curve coefficients for that field, and the model would fail.

- Well depth criteria

  Two options are given for controlling the step size and upper limit of the CDF for well depth:

  - `opt$max.well.depth`

    This option sets the upper limit of `cdf.depth.ow` and `cdf.depth.gw`. The longest total measured well length in the Uinta Basin at the time this manual was written was approximately 65e3 ft long. A much smaller maximum well length of 20e3 ft is recommended because (a) the maximum value is a large outlier, and (b) several regression functions which are fit to well depth (e.g. well capital cost, drilling water usage, etc.) do not have any data points above 20e3 ft, and thus allowing wells to be that long would require significant extrapolation from the fitted data set.

  - `opt$well.depth.step`

    The step size at which to estimate the CDFs for well depth. Any value may be selected here so long as the resulting range between the minimum well depth set in the global options section and the maximum well depth can be divided by the step size to give a whole number.

5.2.3 `EIApriceUpdate`

The only option for this function, `opt$EP.CPI.basis`, is a specification of the CPI value associated with the prices in *.csv file processed by `EIApriceUpdate`. The value for CPI used in that spreadsheet must be the same as the value for CPI entered here.

5.2.4 `leaseOpCostUpdate`

Unless the lease cost report is revived by EIA, new data for this function will most likely not be produced and therefore the user will not need to change any of the options listed here. The three options available are:

- `opt$LU.tstart`

  The starting date to use for selecting lease cost data. The first data point available is for 1994. It is recommended that the full data range be used for this function, therefore this value should not be changed.

- `opt$LU.tstart`

  The stop date to use for selecting lease cost data. The last data point available is for 2009. Again, it is recommended to use the full range of data for this function and therefore this option should not be changed.

- `opt$LOCbasis`

  This options specifies the CPI value to which costs have been inflation adjusted in the *.csv file analyzed by `leaseOpCostUpdate`.

5.2.5 drillingModelUpdate

The only options to select for this data analysis function are for specifying the start (opt$DMU.tstart) and stop (opt$DMU.tstop) points for the time range which will be used for fitting the drilling schedule models (Equations (3.3)-(3.6)). Through trial and error, we have found that there is little correlation between energy prices and drilling rates prior to 1999. Therefore we recommend setting the value of opt$DMU.tstart to "1999-01-01." The value of opt$DMU.tstop is set by default to the global training stop time opt$train.tstop. If there is a substantial change in the drilling behavior of the oil and gas industry that results in poor performance of the drilling schedule models, then the user is encouraged to try different time periods for fitting Equations (3.3)-(3.6).

5.2.6 ARIMAfitUpdate

The options for this function are as follows:

- opt$ARIMA.tstart - start date for ARIMA model fitting. This should always be set to the beginning date of the historical energy price data (1977-07-01).
- opt$ARIMA.tstop - end date for the ARIMA model fitting. This value should either be the last date of the historical price data (if running the model in prediction mode) or the last date in the training data (if running the model in cross-validation mode). For example, if the cross-validating model performance over a twenty year period and price data extended from July 1977 to Dec. 2015, then the end date should be specified as Dec. 1995. The ARIMA model(s) would then be trained on data between July 1977 and Dec. 1995, and the model can be tested against data from Jan. 1996 - Dec. 2015.
- opt$ARIMA.opdq - list object for the manually specified ARIMA oil price model. The list contains two elements: "order," which specifies the ARIMA model order according to the (p,d,q) formula (see Section 3.6), and "ic," which should be a logical TRUE/FALSE value indicating whether or not to include a constant in the ARIMA model.
- opt$ARIMA.gpdq - same as previous input option, but for gas prices.
- opt$ARIMA.plotFit - TRUE/FALSE flag indicating whether or not to plot the cross-validated forecasts for oil and gas from the manually and automitcally fitted ARIMA models. This item is set by default to "TRUE" if opt$crossvalid is set to true, and false otherwise.

5.2.7 EIAforecastUpdate

The EIA Annual Energy Outlook forecast for the simulation period should be entered in this segment of IO_options. The forecast is specified by the following vectors:

- year - vector of dates in YYYY-MM-DD format, should be set so that there is one date for each year in the EIA forecast with the month set to the midpoint (June) of each year.

- `oil` / `loil` / `hoil` - vector of EIA reference / low / high oil prices.
- `gas` / `lgas` / `hgas` - vector of EIA reference / low / high gas prices.

These vectors are combined into the data.frame `opt$forecast`. In addition the forecast values mentioned, the CPI value of the price forecast should also be entered in `opt$EIAbasis`.

5.2.8 `EIAerrorUpdate`

`opt$EEU.tsteps` controls how many years of relative error CDFs to produce. This option should be set to have the same number of years as there are annual oil/gas prices in the EIA forecasts.

5.2.9 `DCAupdate`

This section contains all of the options for controlling the `DCAupdate` function. Options include:

- General DCA fitting options

  These options primarily control the DCA binning algorithm.

  - `opt$minProdRec`

    This option specifies how many non-zero production records a decline curve segment must contain in order to attempt fitting with the decline curve equations. The intended purpose of this option is to prevent over-fitting wells with too few production records. The default and recommended value for this option is 12 (i.e. one year of non-zero production records).

  - `opt$minDayProd`

    This option specifies how many days a well must be in operation in order to include production records from that reporting period as a data point in the DCA. The intended purpose of this option is to prevent the inclusion of production records from wells that have been either shut-in or partially operating in a given reporting period. The recommended value for this term is 28 days.

  - `opt$diff.bin.cutoff`

    This term specifies how large the differential between adjacent production bins must be after normalization in order for a decline curve stop point to be identified in the `binStartStop` function. Small values will result in higher sensitivity but also increased rates of falsely identified stop points. This value should be adjusted in conjunction with `opt$bin`. Combined with the a value for `opt$bin` = 12 months, the recommended value for `opt$diff.bin.cutoff` is 0.15.

  - `opt$bin`

    This option specifies the size (in months) of time window or "bin" for aggregating production volumes in the `binStartStop` function. Larger values will result in more smoothing of the production record, smaller values will give increased sensitivity. This option should be adjusted in

conjunction with `opt$diff.bin.cutoff`. Combined with the a value for `opt$diff.bin.cutoff` = 0.15, the recommended value for `opt$bin` is 12 months.

- `opt$DCAplot`

  Logical TRUE/FALSE flag indicating whether or not the `DCAupdate` function should plot the DCA fits for each well by product type (oil or gas) and field.

- `opt$n.stopB.min`

  This term specifies the first possible bin which will be considered a true stop point. For example, at the default and recommended value of 4, any stop point found in bins 1-3 (years 1-3 with the default bin size of 12 months) will be ignored. The intended purpose of this option is to prevent attempts at fitting distinct decline curves to very early well production histories (which tend to be much more erratic than later production records).

- `opt$n.startT.search`

  This term controls how many top production points should be compared when determining the start point of each production curve. At the default and recommended value of 3, the top 3 largest production records are compared, and the earliest occurring record in that top three is selected as the start point. The intended purpose of this option is to prevent the selection of large production records as the start point of decline curves if those values occur much later in the decline curve segment than other large values (there are occasionally outliers in production records where the largest record occurs in the last third of the decline curve).

- Time step options

  The options `opt$DCA.tstart` and `opt$DCA.tstop` specify that start and stop points of the time range which will be included in the DCA. By default, these values are set to the same range as the global training time range.

- Hyperbolic decline curve options

  This set of options controls the behavior of the nonlinear solver `nlsLM` for fitting Equation (3.13). Options include:

  - `opt$b.start.oil` and `opt$b.start.gas`

    Initial guess for the value of b in Equation (3.13). Trial and error has shown that good initial guesses are 1.78 for oil and 1.32 for gas.

  - `opt$Di.start.oil` and `opt$Di.start.gas`

    Initial guess for the value of D_i in Equation (3.13). Trial and error has shown that good initial guesses are 1.16 for oil and 0.24 for gas.

  - `opt$lower.oil` and `opt$lower.gas`

    Lower limits for the nonlinear solver for each coefficient in Equation (3.13). Limits are given as a vector, corresponding to the order [q_o, b, D_i]. The lower limit for all coefficients should be set to zero.

  - `opt$upper.oil` and `opt$upper.gas`

    Upper limits for the nonlinear solver for each coefficient in Equation (3.13). Limits are given as a vector, corresponding to the order [q_o, b, D_i]. The upper limit for q_o and D_i should be set to ∞ and the upper limit for b should be set to 10.

- Cumulative decline curve options

  This set of options controls the behavior of the nonlinear solver `nlsLM` for fitting Equation (3.14). Options include:

  - `opt$Cp.start.oil` and `opt$Cp.start.gas`

    Initial guess for the value of C_p in Equation (3.14). Trial and error has shown that good initial guesses are 10^3 for oil and 10^4 for gas.

  - `opt$c1.start.oil` and `opt$c1.start.gas`

    Initial guess for the value of c_1 in Equation (3.14). Trial and error has shown that good initial guesses are 0 for both oil and gas.

  - `opt$Qlower.oil` and `opt$Qlower.gas`

    Lower limits for the nonlinear solver for each coefficient in Equation (3.14). Limits are given as a vector, corresponding to the order [C_p, c_1]. The lower limit for C_p should be set to zero, while the lower limit for c_1 should be set to -∞.

  - `opt$Qupper.oil` and `opt$Qupper.gas`

    Upper limits for the nonlinear solver for each coefficient in Equation (3.14). Limits are given as a vector, corresponding to the order [C_p, c_1]. The upper limit for both coefficients should be set to ∞.

5.2.10 `DCAupdateCDF`

The options for this function control the generation of CDFs for each coefficient in Equations (3.13) and (3.14) as well as the CDFs for the time delay in oil and gas production. The available options are:

- `opt$DCA.CDF.type`

  This option controls which CDF generation function is used. Valid options are "Density," which sets the function to use `CDFd`, or "Quantile," which sets the function to use `CDFq`.

- Time step options

  Specifies the time range of DCA data to include in the CDF generation. Both `opt$DCAcdf.tstart` and `opt$DCAcdf.tstop` are set by default to use the global training time range.

- Hyperbolic decline curve CDF generation options

  These options control the upper and lower limits of DCA coefficient values that are included in the DCA CDF generation for the hyperbolic decline curve equation. The specific options are:

  - `opt$cdf.oil.from` and `opt$cdf.gas.from`

    This option specifies the lower limit for the CDFs for oil and gas for the hyperbolic decline curve coefficients and the time delay for oil and gas production. The limits are given in a vector with the order [q_o, b, D_i, t_{delay}]. The lower limit for all terms for both oil and gas should be set to zero.

- `opt$cdf.oil.to` and `opt$cdf.gas.to`

  This option specifies the upper limit for the CDFs for oil and gas for the hyperbolic decline curve coefficients and the time delay for oil and gas production. The limits are given in a vector with the order [q_o, b, D_i, t_{delay}]. The upper limit for all terms except q_o should be set to ∞ for both oil and gas. The upper limit for q_o should be set to 2e3 for oil and 4e4 for gas (based on trial and error with the reasonable range of results for DCA fits to the hyperbolic decline curve equation).

- `opt$cdf.oil.np` and `opt$cdf.gas.np`

  This option specifies how many points to estimate the CDF at if the CDF generation method is set to "Density" (i.e. to use `CDFd`). The order of the vector here is the same as in the previous options ([q_o, b, D_i, t_{delay}]). The values selected by default at set so that the size of the steps in each CDF generated by `CDFd` would have values between 0.1 and 1.

- Cumulative decline curve CDF generation options

  These options control the upper and lower limits of DCA coefficient values that are included in the DCA CDF generation for the cumulative production equation. The meaning of all terms is the same as in the hyperbolic options. The specific options are:

  - `opt$Q.cdf.oil.from` and `opt$Q.cdf.gas.from`

    This option specifies the lower limit for the CDFs for oil and gas for the cumulative decline curve coefficients. The limits are given in a vector with the order [C_p, c_1]. The lower limit for C_p for both oil and gas should be set to zero. The recommended lower limits for c_1 are -30e3 for oil and -400e3 for gas.

  - `opt$Q.cdf.oil.to` and `opt$Q.cdf.gas.to`

    This option specifies the upper limit for the CDFs for oil and gas for the cumulative decline curve coefficients. The limits are given in a vector with the order [C_p, c_1]. The recommended upper limit for C_p is 30e3 for oil and 200e3 for gas. The recommended upper limits for c_1 are 30e3 for oil and 100e3 for gas.

  - `opt$Q.cdf.oil.np` and `opt$Q.cdf.gas.np`

    This option specifies how many points to estimate the CDF at if the CDF generation method is set to "Density" (i.e. to use `CDFd`). The order of the vector here is the same as in the previous options ([C_p, c_1]). The values selected by default at set so that the size of the steps in each CDF generated by `CDFd` would have values between 0.1 and 1.

5.2.11 `reworkUpdate`

The first two options for this update function, `opt$RWU.tstart` and `opt$RWU.tstop` control the time range of data used for generating the rework CDFs. By default, the global training time period values are used. The remaining option, `opt$wc.min` specifies the minimum number of wells that must be left in the well population in order to estimate the rework CDF. The intended purpose of this option is to stop estimating the CDF if the population of existing wells becomes too small to be representative. Without a lower population limit it's also possible to have CDF values for reworking greater than 1. For example, if the oldest well on record (population size 1) were reworked after n months and that well was also n months old, then the probability of

a well being reworked in month n would be 100%. When that probability is added to the previous probability values to get the CDF, the maximum CDF value would be equal to 1 plus the cumulative sum of all other rework probabilities. The recommended value for `opt$wc.min` is 100 wells.

5.2.12 DCA Coefficient Distribution Fitting

The options for this function are:

- `opt$DFmin.rec.count`

  This option specifies the minimum number of DCA fits in a given year (for the entire Basin or each individual field) that must be present in order to attempt fitting the log-normal distribution to points in that year. The intended purpose of this option is to prevent over-fitting when too few data points are available. At the Basin-scale this requirement is almost always satisfied, but on the field-level there are many years (especially for smaller fields) that only have a handful of wells drilled. The recommended value for this option is 10 DCA fits.

- `opt$DFplot.flag`

  This option is a logical TRUE/FALSE flag that specifies whether or not the DCA coefficient distribution fits and regression plots to log-normal distribution parameters should be plotted. Specify as true to output plots.

- Time options

  The start and stop time options for `DCAlnormUpdate` are given as years and are specified independent of the global training time ranges. The accuracy of the log-normal distribution fitting method is highly dependent on the time period selected for fitting (see Section 3.13). Cross-validation tests of the 2010-2014 time period have shown that the best time period to select for matching production data during that test period are `opt$DF.tstart` = 1999 and `opt$DF.tstop` = 2009 (see Figure 3.5). Assuming that recent trends continue, its recommended that `opt$DF.tstart` be left at 1999 and that `opt$DF.tstop` be adjusted to the last year for which data is available in the training period.

5.3 Monte-Carlo Simulation

The options in the MC simulation section of `IO_options` primarily provide character switches that control the calculation methods in the simulation for determining energy prices, the drilling schedule, etc. The options for each function in the MC simulation are presented below.

5.3.1 Energy Price Path Simulation

The option `opt$epf.type` is a character string specifying which method to use for generating energy prices paths as discussed in Section 4.2. Valid options are:

- "a" - for auto-ARIMA price path simulation
- "b" - for manually-specified ARIMA price path simulation
- "c" - for EIA Annual Energy Outlook forecasts with error propagation
- "d" - for using the actual price path
- "e" - for using the constant average price assumption
- "f" - for user specified price paths

The actual energy price paths can only be used if the model is being run in cross-validation mode, and is intended to test the accuracy of the drilling schedule models. The recommended option to use depends on the length of the forecast. For short term forecasts ($\leq$ 1 year), option "e" or "f" is recommnded. For forecasts over the time period $1 < t \leq 5$ years option "c" is recommended. Finally, if the forecasting period is going to be > 5 years, option "a" or "b" is recommended.

In addition the character switch specifying which simulation method to use, the user also needs to specify the following options:

- `opt$epf.EIA.type`

  Character string specifying whether the model should use direct relative error ("direct", see Section 3.8) or fractional relative error ("frac", see Section 3.9). The "frac" option is recommended.

- `opt$epf.EIA.fracProb`

  Probability value between 0 and 1 specifying whether prices should be adjusted up or down by the fractional relative error method. Higher values increase the probability of returning a value of 1 from the binomial distribution (and consequently increase the probability of adjusting energy price paths upwards using Equation 3.11), while lower probability values do the opposite (more likely to return a value of 0 and apply Equation 3.12 to the energy price paths, adjusting them downwards).

  By default, this value is set to 0.5 (equal probability of adjusting upwards/downwards), and should only be changed if the user has reason to believe that the error in EIA's forecasts are skewed in a particular direction.

- `opt$epf.const` - list object with the number of months over which to average prices for oil ("noil") and gas ("ngas") if using the constant average price method. The recommended value is 12 (i.e. one year) for both oil and gas.

- `opt$epf.uepf`

  User specified price paths for oil and gas. These matrices must have one row for each MC iteration and one column for each time step. The values in each element should be in units of $/bbl for oil and $/MCF for gas.

5.3.2 Drilling Simulation

The options to set for the drilling schedule simulation function `drillsim` are:

- `opt$DStype`

  Character string used as a switch to specify what type of method to use for the drilling schedule. Valid options are:

  - "sim" - simulate the drilling schedule using one of the distributed-lag energy price models (see Section 3.5).
  - "actual" - use the actual drilling schedule over the forecasting period. This method can only be used when the model is being run in cross-validation mode, and is intended to test the accuracy of the production simulation calculations.
  - "user" - use a user-specified drilling schedule. This method is intended to serve the same purpose as the user-specified energy price modeling method, but it allows the user to fix the drilling schedule directly (without going through the distributed-lag energy price modeling process).

- `opt$DSsimtype`

  Character string used as a switch to specify which drilling model (Equations (3.3)-(3.6)) to use to predict the drilling schedule. Based on model cross-validation results to the 2010-2014 period, the most accurate drilling model is the oil price model (Equation (3.5)), or option "c." Other options are the prior well model (Equation (3.3), option "a"), energy price model (Equation (3.4), option "b"), and gas price model (Equation (3.6), option "d").

- `DS.uspec`

  If using the user-specifed drilling schedule option, enter the desired schedule here. The schedule must contain only integer values and be in the form of a matrix with rows = MC iterations and columns = forecasting time steps.

5.3.3 Prior Production

The only option to specify for prior wells is the limit on how old a prior well without a fit can be and still be included in the population of "skipped" wells. Most prior wells that don't have fits are wells that had fewer than the minimum number of non-zero production records required by `opt$minProdRec`. The majority of these wells are wells that have been in operation for less than a year. However there are some skipped wells which are much older, and were simply not operated as oil or gas wells for very long. `opt$tend.cut` specifies the upper limit for how old a well can be and still be considered a recently drilled skipped well. The recommended value for this term is 60 months.

5.3.4 Well Data Simulation

The only step in generating `wsim` and `wpri` which requires an input option is a character switch specifying what method should be used for generating decline curve coefficients. The switches are `opt$mc.DCCpick.type.oil` for oil and `opt$mc.DCCpick.type.gas` for gas. If hyperbolic decline curves are used (option "a"), both switches must be set to that option. If one of the cumulative production curve methods is used (options "b" - "d") then the switches can be set independently for oil and gas. Based on the work discussed in Section 3.13, the recommended values are "c" for oil and "b" for gas.

5.3.5 Production Simulation

The user specifies what type of decline curve equation to us with `opt$mc.DCeq.type`. Option "a" will use the hyperbolic decline curve (Equation (3.13)). Option "b" will use the cumulative production curve (Equation (3.14).

5.3.6 Activity-Based Emission Factors

This section of `IO_options` is used to specify all of the mean and standard deviations for each emission factor for each pollutant tracked in the model (CO_2E, CH_4, and VOCs). The specified means and standard deviations are restructured to generate the data.frame `opt$EF`, which is then passed to the function `sim_EF` (which picks the emission factors for each well). The mean and standard deviation for each emission factor and pollutant is given in Table 5.1, which is derived from the literature survey of emission factors in Wilkey et al. [2016].

Table 5.1: Mean (μ) and standard deviation (σ) of emission factors by pollutant type and activity category. All values are in metric tons (10^3 kg) per the unit specified. "ToM" stands for transport of materials.

Activity	10^3 kg/	μ_{CO_2}	σ_{CO_2}	μ_{CH_4}	σ_{CH_4}	μ_{VOC}	σ_{VOC}
Site Preparation	well	2.08E+2	7.90E+1	9.90E+0	3.37E+0	1.58E+0	6.00E-1
ToM for Drilling	well	4.00E-1	5.60E-1	8.60E-6	1.22E-5	1.38E-6	1.95E-6
ToM for Completion	well	2.10E-1	2.90E-1	4.36E-6	6.16E-6	6.97E-7	9.86E-7
ToM for Rework	well	3.05E+0	4.31E+0	7.71E-5	1.01E-4	1.15E-5	1.62E-5
ToM for Production	well	1.36E+0	1.93E+0	3.29E-5	4.65E-5	5.26E-6	7.43E-6
Completion	well	1.94E+3	9.67E+2	9.24E+1	4.60E+1	1.48E+1	7.37E+0
Gas Production	well/year	3.58E+0	3.33E+0	1.73E-1	1.58E-1	6.50E-2	6.08E-2
Gas Processing	MCF gas	9.01E-4	4.60E-5	5.58E-6	3.91E-6	8.90E-7	6.20E-7
Gas Transmission	MCF gas	4.18E-3	3.42E-3	1.99E-4	1.63E-4	3.18E-5	2.60E-5
Oil Production	bbl oil	4.91E-5	4.91E-5	2.34E-6	2.34E-6	8.88E-7	8.88E-7
Oil Transport	bbl oil	1.15E-3	1.15E-3	2.82E-7	2.82E-7	3.84E-7	3.84E-7

In addition to generating `opt$EF`, this section also specifies emissions reductions in the dataframe `opt$EFred`. Each emissions reduction must be programmed into the `Ecalc` function manually. Any new reductions added to this section can be coded following the examples of those currently programmed in the `Ecalc` function.

5.4 Post-Processing

5.4.1 Plots

These options control what plots are created by the `postProcess` function and how they are labeled. The specific options are:

- `opt$exportFlag`

  Logical TRUE/FALSE value indicating whether or not plots generated by `postProcess` are to be exported to *.pdf (if true), or printed to the user's screen (if false).

- `opt$exportSingle`

  Logical TRUE/FALSE value indicating whether or not to combine all plots into a single PDF. If true, one PDF file will be created; otherwise each plot will be created as a separate PDF.

- `opt$prefix` and `opt$affix`

  These two options control the beginning and end of any *.pdf files that are generated by `postProcess` if `opt$exportFlag` is set to true. The naming system used is:

  `[prefix][file name][affix].pdf`

  where "prefix" and "affix" are any character string that is specified by each of these options, respectively. Any naming convention can be used.

- `opt$plist`

  This data.frame contains a two columns with one row for every set of plots programmed into `postProcess`. The first column specifies the name associated with the plot(s) in question as a character string, and is inserted into the "file name" segment of the *.pdf naming convention noted above. The second column is a logical TRUE/FALSE value indicating whether or not the plot(s) should be generated. Some rows control multiple plots. For example, row 28, which controls water balance results, generates nine separate plots (but all are printed out to the same file). Note that the order of the plots is important. As the `postProcess` function checks whether or not each plot is going to be printed a counter variable "j" is incremented after each if-statement. Therefore any new plots that are added to `postProcess` should be added at the bottom of both that script and `opt$plist`.

- `opt$defFontSize`

  This option specifies how large the font size in plots should be relative to the default size of 1.00 (e.g. a value of 1.50 is 50% larger than the default). For improving readability of plots in presentations and other media, the default value of this term has been set to 1.25.

- `opt$quant`

  This vector specifies what quantiles (e.g. percentiles) to include in the plot results. For example, specifying [0.1, 0.25, 0.5, 0.75, 0.9] would calculate the 10^{th}, 25^{th}, 50^{th}, 75^{th}, and 90^{th} percentiles of each simulation result in all plots designed to show those results.

5.4.2 Excel Export

This section controls the options for exporting the equipment-based emission results to an Excel spreadsheet. The options are:

- `opt$xls.export`

  Logical `TRUE/FALSE` flag indicating whether or not the emission results should be exported to Excel.

- `opt$xls.name`

  Character string specifying the name of the results spreadsheet. The version number specified by `opt$file_ver` will automatically be appended to the end of this character string, along with the appropriate file type extension (*.xlsx).

Part IV

EF_options

Chapter 6

EF_options

`EF_options` is the third main script in the model, and contains all of the user input options for incorporating equipment-based emissions into the model. The script contains one section for global options, followed by sections for each type of equipment (well completions, RICE and turbines, etc.).

6.1 Global Options

The first section of `EF_options` sets just one global option, namely it creates the `eopt` list object with the line:

```
eopt <- NULL
```

to which all subsequent options in `EF_options` are added. This line must be present and should not be changed.

6.2 Equipment Type Options

All of the equipment related sections of `EF_options` have the following format:

- Emission factors

  The following types of equipment use emission factors that are specified by the user:

  - Well completions (`eopt$wc.EF`)
  - Separators and heaters (`eopt$sh.EF`)
  - Dehydrators (`eopt$dh.EF`)
  - Tanks (`eopt$tank.EF`)

- Pneumatic controllers (`eopt$pctrl.EF`)
- Fugitive emissions (`eopt$fug.EF`)

By default all of these emission factors have been set to match those given in UDAQ's OGEI database spreadsheet. The emission factors used for all other types of equipment are specified directly by the entries in the CPTs created by the `emissionUpdate` function.

- Emission reduction options

  Every type of equipment has a built-in set of reduction options labeled `eopt$r$...` where "..." is replaced with the following abbreviations:

  - wc - well completion
  - rt - RICE and turbines
  - sh - separators and heaters
  - dh - dehydrators
  - tank - tanks
  - truck - truck loading
  - pctrl - pneumatic controllers
  - ppump - pneumatic pumps
  - fug - fugitive emissions

  The emission reduction options that can be set are:

  1. `tDrill`

     The time step in which a well is drilled. No wells drilled prior to the specified calendar date will be considered for emission reductions. The calendar dates specified in YYYY-MM-DD format will be converted into MC simulation time steps by the `calTstep` function described in Section 2.2.6.

  2. `tstep`

     Similar to `tDrill`, this input specifies the date at which the emission reduction will be applied. Only calendar dates on and after the date specified will be reduced. For example, if the MC simulation was 60 months long, and the 30^{th} month during the simulation period was specified as `tstep`, then only the 30^{th} - 60^{th} columns of any applicable rows would have their emissions reduced.

  3. `wellType`

     Character string indicating the type of wells to which reductions may be applied. Valid options are "OW" for oil wells only, "GW" for gas wells only, and "all" for all well types.

  4. `juris`

     Character string indicating the jurisdiction to which reductions may be applied. Valid options are:

     - "federal" for federal jurisdiction only

- "state" for state jurisdiction only
- "fee" for fee jurisdiction only
- "indian" for Indian jurisdiction only
- "all" for all jurisdiction types

5. `county`

   Character string indicating the county to which reductions may be applied. Valid options are "UINTAH" for Uintah county only, "DUCHESNE" for Duchesne county only, or "all" for any county.

6. `a...`

   Threshold annual emissions of each applicable pollutant species [...] (PM_{10}, VOC, etc.). Only those wells which have an annual average emission rate $\geq$ the specified amount (in ton/yr) will considered for emission reductions. Each type of equipment lists only the pollutants it generates in this section.

7. `moil` and `mgas`

   The maximum oil (`moil`) and gas (`mgas`) production rate. Only those wells with values $\geq$ the values specified will be considered for emission reductions

   Note that care should be used when applying this criteria, as many wells in the simulation will have no production in their first month of operation (due to time delays between well completion and first production). If this criteria is applied, the reduced emissions results will appear to spike at the end of the forecasting period (especially well completion emissions). Therefore it is recommended that this criteria should not be applied to well completions (i.e. `moil` and `mgas` should be set to 0 for well completions).

8. `red...`

   Finally, the actual emission reductions to apply for each applicable species [...]. The values listed here should be expressed as fractions (e.g. 0.9 for a 90% reduction). Each type of equipment lists only the pollutants it generates in this section.

Part V

postProcess

Chapter 7

Post-Processing

After the MC simulation is complete there are a number of post-processing options that can be used to visualize and export the results. A set of predefined plots are provided in the script `postProcess` which runs at the end of `main`. The structure of `postProcess` is discussed below.

7.1 Queries and Calculations in `postProcess`

If the model is being run in cross-validation mode, the `postProcess` script begins by determining the actual values of several terms during the test period, including: energy prices, drilling schedule, and oil and gas production (from new wells, prior wells, and all wells). If the model is being run in predictive mode, there is no actual data to compare against and this step is skipped.

The next step in the post-processing analysis is to summarize the full range of results for all the MC simulation runs for each term of interest. The method we have utilized is to calculate the quantiles (i.e. percentiles) of the results for each term as a function of time using the `quantile` function. The percentiles sampled are set by `opt$quant` (by default, the 10^{th}, 30^{th}, 50^{th}, 70^{th}, and 90^{th} percentiles). These percentiles are calculated for nearly every term in the MC simulation (energy prices, drilling schedule, oil and gas production, emissions, etc.). The calculated quantiles are all stored in matrices with one row for each quantile calculated and one column for each time step. The quantile matrix for each term uses the same name as the MC simulation results matrix it is based on, with an additional ".q" placed on the end of the matrix name. For example, the simulated drilling schedule for each simulation run is stored in the matrix `Drilled`, and the quantiles of that drilling schedule are stored in `Drilled.q`.

7.2 Excel Export

The `postProcess` script includes the option to export equipment-based emission calculation results to an Excel spreadsheet. The exported spreadsheet contains one worksheet for (a) total emissions by species and

(b) emissions by equipment type and species. Both types of results are given with one row for each time step in the MC simulation and one column for each quantile in `opt$quant`. Additionally, separate worksheets are created for both the base and reduced emissions.

7.3 Plots in `postProcess`

Most of the predefined plots share the same overall structure. The plots begins with an if-statement that controls whether or not the plot is generated (based on user input in `IO_options`). If the plot is going to be generated, a second if-statement controls whether or not that plot will be saved to a *.pdf file (or printed to the user's screen). Next, the main plot function is called, usually to plot the largest quantile result as a function of time (e.g. the 90^{th} percentile of oil price forecasts). The first plot call also creates the main elements of the plot (axes, labels, titles, etc.). Afterwards the `lines` function is used to drawn in the remaining quantile results. If the model is being run in cross-validation mode, the actual value is then drawn. Finally, a legend is inserted. Note that if the number of quantiles is ever changed the legend entries in each plot will also have to be adjusted to match (as all the predefined plots are currently configured to use 10^{th} - 90^{th} percentile range). An example of a typical xy-quantile plot is shown in Figure 7.1

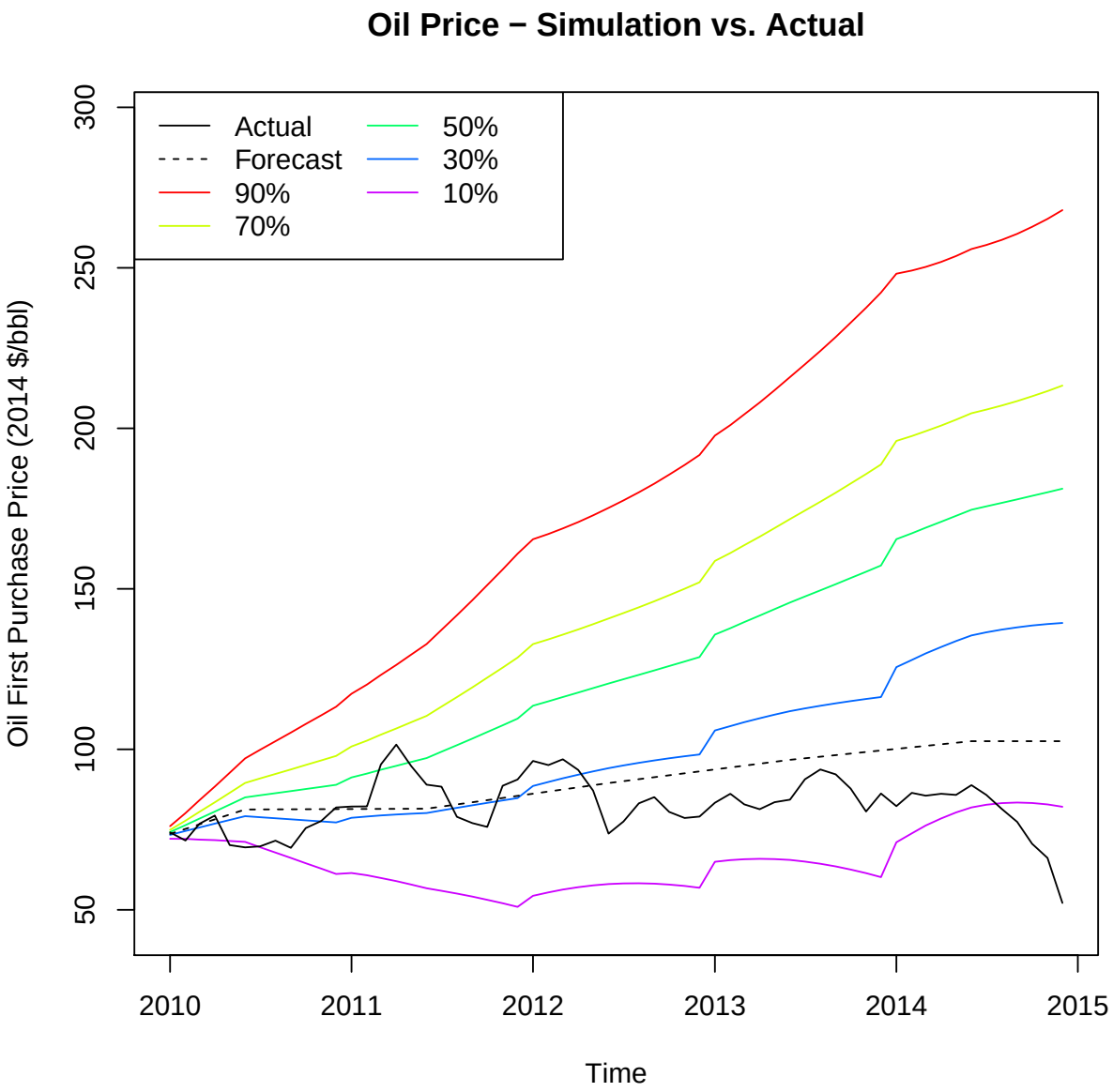

Figure 7.1: Example xy-quantile plot of oil price paths.

7.4 Other Export Options

A number of options exist for exporting results from R to other programs and file formats. See Section 9.5 for a detailed discussion of exporting options.

Part VI

Minor Scripts

Chapter 8

Minor Scripts

In addition to the major scripts discussed previously, the model contains several other minor scripts which are discussed below.

8.1 R-Markdown Reports

Two automated reports have been included in the model under the "Data Analysis" folder of the Git repository. These reports are written in the R-Markdown language, and return as an output an html document. The reports are intended to be used after the user has run the data analysis segment of the model. The two reports generated are:

- `dataAnalysis.Rmd`

  This report summarizes the CPTs produced by the `emissionUpdate` function, returning for each piece of equipment:

  - The inputs from the OGEI database which are used by model
  - The dimensions of the original table(s) in the OGEI database and the size of the CPT
  - The top 10 most common entries in each CPT
  - A summary statistics of each input variable (if they are numeric), or the unique values of the input variable (if they are character strings)
  - A number of exploratory plots for each type of equipment (e.g. CDFs of interesting input variables).

- `tanksDA.Rmd`

  This report creates a set of exploratory data analysis plots of tank VOC emissions vs. fluid (oil, condensate, and water) throughput volumes for (a) the OGEI database and (b) the ten largest operators in the database (based on the number of entries by each operator in the database). Data points are also differentiated based on whether or not they are from sources with or without controls.

8.2 README.md

This markdown document (located in the home directory of the Git repository with `main`, `IO_options`, etc.) is used to create the README published on the model's Bitbucket webpage. It lists the steps necessary for downloading and running the model, and provides links to the model's supporting documentation (including this User Manual).

Part VII

Using the Model

Chapter 9

Tutorials for Common Tasks

9.1 Model Installation and Setup

The instructions below outline the steps necessary for setting up the model using version control with Bitbucket (`https://bitbucket.org`) and RStudio (`http://www.rstudio.com`). Alternative installation methods exist, however at a minimum, the user must (a) install R and (b) copy the model folder (containing all of the scripts and functions in the model) and its supporting data folder (containing all of the raw and prepared data files) onto their computer.

1. Download and install both R and Rtools (`http://www.r-project.org`).

   (a) Special notes for installing Rtools:

      i. At the first three screens of the installer ("Welcome", "Information", "Select Destination"), page just click "Next".

      ii. At the "Select Components" page choose the default "Package authoring installation" and click "Next".

      iii. At the "Select Additional Tasks" page find choose to let the installer "Edit the system PATH" by clicking a checkbox that appears beside the current value of the PATH.

      iv. At the "System Path" page just click "Next".

      v. At the "Ready to Install" page click "Install".

2. Download and install RStudio (`http://www.rstudio.com/products/rstudio/download`).

3. Download and install Git for version control and set your global username and email following the instructions here: `https://confluence.atlassian.com/bitbucket/set-up-git-744723531.html`.

4. Setup an account on bitbucket.org (`https://bitbucket.org/account/signup/`). If the repository is private, ask the current repository manager to share the repository with your account.

5. Download the model's supporting raw and prepared data files from Dropbox (`https://www.dropbox.com/sh/sfc47tx1c2d9nzw/AADDnBzPnw4S5ywAVfVZ6L1Ya?dl=0`).

6. Launch RStudio.

7. Confirm that RStudio has the correct path to the Git executable. From the menu bar select "Tools → Global Options" and then in the Options menu select "Git/SVN." Enter the filepath to the Git executable in the appropriate dialogue box (by default: [installation directory]/Git/bin/git.exe). If you had to find the Git executable, restart RStudio.

8. From the menu, select "File → New Project". In the dialogue menu that pops up, select "Version Control" then "Git."

9. The repository can be connected to using either HTTPS or SSH.

   (a) To connect using HTTPS, paste the URL for the respository (e.g. `https://wilkey@bitbucket.org/wilkey/ub_o-g_emissions.git`) into the "Repository URL" box. You will then be prompted for your username and password when you connect to the server.

   (b) To connect using SSH, you must first create a public/private RSA ID and upload your public key to Bitbucket following the instructions given in the tutorial here: `https://confluence.atlassian.com/bitbucket/set-up-ssh-for-git-728138079.html`. Then, paste the Git SSH location (e.g. `git@bitbucket.org:wilkey/ub_o-g_emissions.git`) into the "Repository URL" box.

10. Enter the desired name for the project directory (i.e. the name of the folder that will be created on your computer) and the file path for the folder, or accept the RStudio defaults.

11. Click "Create Project" and Git will clone all of the folders and scripts in the repository to your machine.

12. Open the script `main`.

13. Under the section of the script labeled "Paths" change the `pwd.drop` and `pwd.git` paths to the location on your computer where you downloaded the Dropbox folder and where you cloned the repository to, respectively.

14. Install the packages necessary to run `main`. The required packages are listed under the "Packages" section header and in Table 2.1. You can install packages from the menu by selecting "Tools → Install Packages..." or by using the `install.packages` command in the R console.

15. Save the changes made to `main`.

You can now run the script by clicking the "Source" button on in the top-right corner of the source window. By default, source runs without printing to the console, however you can also run source with echo and it will print every line in the script as it is executed.

9.2 Updating the Model with New Data

The following steps should be taken to update the source data used by the model. All data sources are updated on at least an annual basis, however some update more frequently. The user is free to update any data source files as desired using the TRUE/FALSE flags in `IO_options`.

1. Download the most recent UDOGM database files from UDOGM's Data Research Center (`https://oilgas.ogm.utah.gov/Data_Center/DataCenter.cfm#download`). In particular, download the latest copies of:
   - "Well Data" (welldata.exe)
   - "Well History Data" (histdata.exe)
   - "Production Data (ALL Data - too large for Excel!)" (proddata.exe)
   - "Field Data" (fieldata.exe)
2. Run all of the downloaded excutable files, which are self-contained zip archives of *.dbf database files, and paste the *.dbf files into the "Raw Data" folder.
3. Update the "EIA_HistPrices.csv" using the spreadsheet and data sources discussed in Section 3.3.
4. Enter the desired EIA Annual Energy Outlook forecast data in `IO_options` as discussed in Section 5.2.7.
5. Update the EIA error analysis data *.csv files. If using the direct relative error method (see Section 3.8), then the two required *.csv files are "EIA_gp_error_export.csv" and "EIA_op_error_export.csv". If using the fractional relative error method (see Section 3.9), then the two required *.csv files are "EIA AEO frac error gp export.csv" and "EIA AEO frac error op export.csv". Note that this data source can only be updated on an annual basis (as EIA forecast and actual price data becomes available).
6. Extract the following tables from the UDAQ OGEI database as *.csv files and place them in the "OGEI" folder (located under in the "Raw Data" Dropbox folder):
   - apis
   - component_count
   - dehydrators
   - facilities_list
   - fugitives
   - operator_info
   - pneumatic_controllers
   - pneumatic_pumps
   - production_areas
   - rice_emission_factors
   - rice_turbines

- separators_heaters
- tanks
- truck_loading
- well_completions

7. Update any other source data file or input in `IO_options`.
8. If desired (but highly recommended) set `opt$file_ver` to a new version number (e.g. "v9"). This will prevent any previous results from being overwritten, and also allows the user to easily revert back to any prior version of the model's data analysis results.
9. Set the appropriate TRUE/FALSE update flags for each data analysis segment in `IO_options`. If the `opt$file_ver` was changed, set all update flags to TRUE (or manually copy and rename a previously prepared *.rda file in the "Prepared Data" folder to match the new file version name).
10. Run the model by sourcing `main`.

9.3 Running the Model: Cross-Validation

For the purpose of validating and tuning the model's performance, the model can be cross-validating by segmenting the available data into "training" and "testing" time periods in `IO_options`. The model will then run all data analysis functions using only the data from the training period, and the results of predictions made with the training data can be compared to the independent test data set. The steps for running the model in cross-validation mode are outlined below.

1. Set user input options in `IO_options`
   (a) Set global options in `IO_options`.
      i. Set `opt$crossvalid` to TRUE. This will enable post-processing analysis steps aimed at making comparisons between test data and model predictions.
      ii. Enter a date for `opt$tstart` and `opt$tstop` that is fully within the date range where data is available. This date range will become the time period over which the model simulation is run, and over which the model is compared to the test data. Using the last five years of data is recommended, or the last "n" years where "n" is the length of time for which you would like to make predictions into the future.
      iii. Enter a date for `opt$train.tstart` and `opt$train.tstop` that is fully within the date range where data is available and that does not overlap the test period `opt$tstart - opt$tstop` time range. In general, only data inside of this time range will be used in the data analysis section of the model.
      iv. If desired, set a new `opt$file_ver` version number.
      v. If desired, set the data analysis functions you wish to be run to TRUE. If a new file version number has been set, run all of the update functions.

(b) Set the desired options for all data analysis update functions. The meanings of all data analysis update options are discussed in Section 5.2. Several data analysis update options are discussed below.

i. `leaseOpCostUpdate` Options
The start and stop times steps for the lease cost update function are for the entire time range where data is available. Given that there is no independent test data to compare against, this value should be left as-is to cover the full data set.

ii. `drillingModelUpdate` Options
The start and stop time steps are also set independently for this function. Past efforts at fitting drilling rates as a function of energy prices have shown that there is little to no correlation between energy prices and drilling in the Uinta Basin prior to the late 1990s. Going forward, the start date of the drilling update function (`opt$DMU.tstart`) should be set no earlier than 1999-01-01, and the end date should be no greater than `opt$tstart`. Aside from those limits, the user should set the time span for the drilling model update function that they believe most closely reflects current and/or future market conditions in the Uinta Basin.

iii. `EIAforecastUpdate` Options
Enter the EIA AEO forecast for Rocky Mountain wellhead oil and gas prices for the same year as `opt$tstart` (e.g. if `opt$tstart` = 2010-01-01, then enter the 2010 Annual Energy Outlook forecast). Enter as many years of forecasted prices as are covered by the test period time span between `opt$tstart` - `opt$tstop`.

iv. DCA Coefficient Distribution Fitting Options
This update function also uses an independent training time range. Given the trends in DCA fitted distribution coefficient values, selecting a start date of $<$ 1999 is not recommended, and the stop date should be less than the value of `opt$tstart`.

(c) Set desired Monte-Carlo simulation options.

(d) Set desired post-processing options. The predefined post-processing plots are primarily designed to enable comparisons between model predictions and the test period data.

2. Set desired equipment-based emission calculation options in `EF_options`.

3. Run the model by sourcing `main`.

9.4 Running the Model: Prediction

Running the model in predictive mode involves many of the same steps as cross-validation, except that the training period is extended to cover the entire range of time for which data is available and `opt$tstart` - `opt$tstop` is pushed forward into a future time period. The steps for running the model in predictive mode our outlined below.

1. Set user input options in `IO_options`

(a) Set global options in `IO_options`.

i. Set `opt$crossvalid` to FALSE. This will prevent post-processing errors by omitting analysis steps aimed at making comparisons between test data and model predictions.

ii. Enter a date for `opt$tstart` and `opt$tstop` that begins one month into the future and extends for any duration of time. For example if data was available up to 2015-08-01, then the value of `opt$tstart` should be set to 2015-09-01. The start date of the simulation period shouldn't be set any further out than one month into the future because many steps in the model rely on knowing the actual value of an input parameter just prior to the start of the simulation period (e.g. oil/gas first purchase prices, initial wells drilled, etc.). This date range will become the time period over which the model simulation is run.

A simulation period of five years is recommended. Model predictions become increasingly uncertain with time, especially in regards to energy price paths. If longer simulation periods are desired, the only limitation is the time period for which energy prices are available. If the EIA AEO error propagated forecasted is being used, the upper limit is the number of years into the future for which forecasted prices are available (at present, EIA AEO forecasts extend to 2040). If the GBM forecasting method is being used, there is no limit to how far into the future prices can be estimated.

iii. Enter a date for `opt$train.tstart` and `opt$train.tstop` that is fully within the date range where data is available and that does not overlap the test period `opt$tstart` - `opt$tstop` time range. For the purposes of running the model in predictive mode, it's recommended that all available data be covered in the training period.

iv. If desired, set a new `opt$file_ver` version number.

v. If desired, set the data analysis functions you wish to be run to TRUE. If a new file version number has been set, run all of the update functions.

(b) Set the desired options for all data analysis update functions. The meanings of all data analysis update options are discussed in Section 5.2. Several data analysis update options are discussed below.

i. `leaseOpCostUpdate` Options

The start and stop times steps for the lease cost update function are for the entire time range where data is available. Given that there is no independent test data to compare against, this value should be left as-is to cover the full data set.

ii. `drillingModelUpdate` Options

The start and stop time steps are also set independently for this function. Past efforts at fitting drilling rates as a function of energy prices have shown that there is little to no correlation between energy prices and drilling in the Uinta Basin prior to the late 1990s. Going forward, the start date of the drilling update function (`opt$DMU.tstart`) should be set no earlier than 1999-01-01, and the end date should be no greater than `opt$tstart`. Aside from those limits, the user should set the time span for the drilling model update function that they believe most closely reflects current and/or future market conditions in the Uinta Basin.

iii. `EIAforecastUpdate` Options

Enter the EIA forecast for Rocky Mountain wellhead oil and gas prices from EIA's Annual Energy Outlook forecast for the same year as `opt$tstart` (e.g. if `opt$tstart` $\bar{2}$015-01-01, then enter the 2015 Annual Energy Outlook forecast). Enter as many years of forecasted prices as are covered by the test period time span between `opt$tstart` - `opt$tstop`.

iv. DCA Coefficient Distribution Fitting Options
This update function also uses an independent training time range. Given the trends in DCA fitted distribution coefficient values, selecting a start date of < 1999 is not recommended, and the stop date should be less than the value of `opt$tstart`.

(c) Set desired MC simulation options.

(d) Set desired post-processing options. The predefined post-processing plots are primarily designed to enable comparisons between model predictions and the test period data. All prior comparisons will be dropped from prepared plots if `opt$crossvalid` is set to FALSE.

2. Set desired equipment-based emission calculation options in `EF_options`.

3. Run the model by sourcing `main`.

9.5 Exporting Results

After a simulation run is complete there are a variety of data export options available that can be used to extract information from any object in the R workspace. If the data set that you would like to extract is relatively small (i.e. $< 10^3$ elements) a fast export option is to use the `clipboard` function to copy the object to your computer clipboard, from which the copied data can be pasted into any spreadsheet program. If the data set is large, then the next easiest export option is to save the data to a *.csv file using the `write.csv` function. For example, the command:

```
write.csv(osim, file.path(path$data, "osim.csv"))
```

would write to a *.csv file named "osim.csv" in the "Prepared Data" folder the object `osim`, which contains the total oil production records for each MC simulation run. The `postProcess` script also includes options for directly generating Excel spreadsheets with the model's emission results. The code in that script can be used as a template to export other model results to Excel. Finally, there are a number of exports tools in R for connecting to relational databases. Detailed instructions for how to use relational databases (and other data import/export topics) are covered in documentation available from the R Core Team at `https://cran.r-project.org/doc/manuals/r-release/R-data.pdf`.

Bibliography

J. J. Arps. Analysis of decline curves. *Transactions of the American Institute of Mining, ...*, 160(160):228, 1945. URL http://www.pe.tamu.edu/blasingame/data/z_zCourse_Archive/P689_reference_02C/z_P689_02C_ARP_Tech_Papers_(Ref)_(pdf)/SPE_00000_Arps_Decline_Curve_Analysis.pdf.

Rasmus Baath. *beepr: Easily Play Notification Sounds on any Platform*, 2015. URL http://cran.r-project.org/package=beepr.

Rob Carnell. *lhs: Latin Hypercube Samples*, 2012. URL http://cran.r-project.org/package=lhs.

David B Dahl. *xtable: Export tables to LaTeX or HTML*, 2014. URL http://cran.r-project.org/package=xtable.

Marie Laure Delignette-Muller and Christophe Dutang. {fitdistrplus}: An {R} Package for Fitting Distributions. *Journal of Statistical Software*, 64(4):1–34, 2015. URL http://www.jstatsoft.org/v64/i04/.

M Dowle, T Short, S Lianoglou, A Srinivasan with contributions from R Saporta, and E Antonyan. *data.table: Extension of data.frame*, 2014. URL http://cran.r-project.org/package=data.table.

Timur V Elzhov, Katharine M Mullen, Andrej-Nikolai Spiess, and Ben Bolker. *minpack.lm: R interface to the Levenberg-Marquardt nonlinear least-squares algorithm found in MINPACK, plus support for bounds*, 2013. URL http://cran.r-project.org/package=minpack.lm.

G Grothendieck. *sqldf: Perform SQL Selects on R Data Frames*, 2014. URL http://cran.r-project.org/package=sqldf.

RJ Hyndman. forecast: Forecasting functions for time series and linear models, 2016. URL http://github.com/robjhyndman/forecast.

Andrew Landgraf. Copying Data from Excel to R and Back, 2014. URL http://alandgraf.blogspot.com/2013/02/copying-data-from-excel-to-r-and-back_24.html.

Uwe Ligges and Martin Mächler. Scatterplot3d - an R Package for Visualizing Multivariate Data. *Journal of Statistical Software*, 8(11):1–20, 2003. URL http://www.jstatsoft.org.

Gabriel A Lozada and Michael Hogue. The Effect of Proposed 2009 Tax Changes on Utah's Oil and Gas Industry. Technical report, Univ. of Utah Dept. of Economics, Salt Lake City, UT, 2008.

R Core Team. R: A Language and Environment for Statistical Computing, 2015a. URL http://www.r-project.org/.

R Core Team. *foreign: Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, Weka, dBase, ...*, 2015b. URL http://cran.r-project.org/package=foreign.

U.S. Bureau of Labor Statistics. Consumer Price Index, 2015. URL http://www.bls.gov/cpi/.

U.S. Energy Information Administration. Oil and Gas Lease Equipment and Operating Costs 1994 Through 2009. Technical report, U.S. Energy Information Administration, Washington, DC, 2010. URL http://www.eia.gov/pub/oil_gas/natural_gas/data_publications/cost_indices_equipment_production/current/coststudy.html.

U.S. Energy Information Administration. Annual Energy Outlook 2014. Technical report, U.S. Energy Information Administration, Washington, DC, 2014. URL http://www.eia.gov/forecasts/aeo/pdf/0383(2014).pdf.

U.S. Energy Information Administration. Henry Hub Natural Gas Spot Price, 2015a. URL http://www.eia.gov/dnav/ng/hist/rngwhhdM.htm.

U.S. Energy Information Administration. U.S. Natural Gas Wellhead Price, 2015b. URL http://www.eia.gov/dnav/ng/hist/n9190us3M.htm.

U.S. Energy Information Administration. Utah Natural Gas Wellhead Price, 2015c. URL http://www.eia.gov/dnav/ng/hist/na1140_sut_3a.htm.

U.S. Energy Information Administration. Utah Crude Oil First Purchase Price, 2015d. URL http://www.eia.gov/dnav/pet/hist/LeafHandler.ashx?n=pet&s=f004049__3&f=m.

Utah DOGM. Data Research Center, 2015. URL http://oilgas.ogm.utah.gov/Data_Center/DataCenter.cfm.

Alexander Walker. openxlsx: Read, Write and Edit XLSX Files, 2015. URL https://cran.r-project.org/package=openxlsx.

Ian Walton. Shale Gas Production Analysis - Phase 1. Technical report, Energy & Geoscience Institute at the University of Utah, Salt Lake City, UT, 2014.

Hadley Wickham. The Split-Apply-Combine Strategy for Data Analysis. *Journal of Statistical Software*, 40 (1):1–29, 2011. URL http://www.jstatsoft.org/v40/i01/.

Hadley Wickham and Winston Chang. devtools: Tools to Make Developing R Packages Easier, 2016. URL https://cran.r-project.org/package=devtools.

Jonathan Wilkey, Kerry Kelly, Isabel Cristina Jaramillo, Jennifer Spinti, Terry Ring, Michael Hogue, and Donatella Pasqualini. Predicting emissions from oil and gas operations in the Uinta Basin, Utah. *Journal of the Air & Waste Management Association*, 66(5):528–545, may 2016. ISSN 1096-2247. doi: 10.1080/10962247.2016.1153529. URL http://www.tandfonline.com/doi/full/10.1080/10962247.2016.1153529.

Achim Zeileis and Gabor Grothendieck. zoo: S3 Infrastructure for Regular and Irregular Time Series. *Journal of Statistical Software*, 14(6):1–27, 2005. URL http://www.jstatsoft.org/v14/i06/.

Appendices

Appendix A

Nomenclature

A.1 User Manual Nomenclature

Table A.1: User Manual Nomenclature

Symbol/Abbreviation	Description
μ	Percent drift term in GBM equations
σ	Volatility term in GBM equations
AP	Actual Price
b	Decline exponent
C	Cost
c_1	Fitted coefficient in cumulative production equation
C_p	Fitted coefficient in cumulative production equation
D_i	Initial decline rate
dt	R's `difftime` function
E_i	Emissions of species i
EF_{fug}	Fugitive emission factor
EF_i	Emission factor for species i
EP	Energy prices

Symbol/Abbreviation	Description
F	Diesel fuel usage
FP	Forecasted price
GP	Gas price
HD	Heat duty rating
hp	Horsepower
HV	Heating value
I	Index
LC	Lease equipment or operating cost
m_{VOC}	VOC weight fraction
MW	Molecular weight (lb/lbmol)
n_i	Number of high/intermittent/low bleed devices (for pneumatic controllers) or number of emission sources (for fugitive emissions)
OP	Oil price
P	Production rate of oil or gas
p_{oil}	Oil production
P_{vap}	Vapor pressure (in psia)
PP	Predicted Price
PR	Gas production rate
q	Production rate
Q	Cumulative production
q_i	Initial production rate
r	Emission reduction fraction
R	VOC emissions ratio (ton VOC)/(bbl oil)
RE	Relative error
$round$	R's `round` function

Symbol/Abbreviation	Description
S	"S" factor for truck loading
SP	Simulated price
t	Time (various units)
T	Temperature (in °R)
t_{date}	Calendar date of a MC simulation time step
t_{delay}	Time delay between t_{Drill} and the start of production
t_{Drill}	Time step in which a well is drilled
t_{start}	Calendar date of the start of the MC simulation
t_{step}	MC simulation time step
V_{pilot}	Volume of gas used by the combustor pilot
VR	Average pneumatic pump vent rate (CF/minute)
W	Wells drilled
w_{frac}	Well fraction
AEO	Annual Energy Outlook
ARIMA	Auto-regressive integrated moving average
bbl	Barrel (42 US gallons)
Btu	British thermal unit
CDF	Cumulative distribution function
CF	Cubic feet
CH_2O	Formaldehyde
CH_4	Methane
CO	Carbone monoxide
CO_2E	Carbon dioxide equivalence
CPI	Consumer price index

Symbol/Abbreviation	Description
CPT	Cumulative probability table
csv	Comma-separated variable
D	Well depth in ft
DCA	Decline curve analysis
EBE	Equipment-based emissions
EIA	U.S. Energy Information Administration
EPF	Energy price forecast
FPP	First purchase price
gal	Gallon
GBM	Geometric Brownian motion
IDE	Integrated development environment
lb	Pound
lbmol	Pound-mole
LHS	Latin hypercube sampling
LOC	Lease operating cost
MC	Monte-Carlo
MCF	Thousand standard cubic feet
MCFD	Thousand standard cubic feet (MCF) per day
MLE	Maximum likelihood estimation
MMBtu	Million British Thermal Units (Btu)
NO_x	Nitrogen oxides
NSPS	New source perfomance standards
OGEI	Oil and gas emissions inventory
pdf	Portable document format

Symbol/Abbreviation	Description
PDF	Probability density function
PM_{10}	Particulate matter (¡ 10 microns in diameter)
$PM_{2.5}$	Particulate matter (¡ 2.5 microns in diameter)
rda	Compressed file format used by R
RICE	Reciprocating internal combustion engine
RSS	Residual sum of the squares
SCF	Standard cubic feet
SO_x	Sulfur oxides
ToM	Transport of Materials
UDAQ	Utah Division of Air Quality
UDOGM	Utah Division of Oil, Gas and Mining
VOC	Volatile organic compounds
VRU	Vapor recovery unit

A.2 Model Nomenclature

Table A.2: Model Nomenclature

Object	Description
`aE`	Activity based emissions results list
`alinecolor`	Line color used for actual results in `postProcess` plots
`alinetype`	Line type used for actual results in `postProcess` plots
`alinewidth`	Line width used for actual results in `postProcess` plots

Object	Description
all.p	Actual oil and gas production during the simulation period. Calculated in postProcess if the model is being run in cross-validation mode.
aPR	Actual production ratio of oil and gas from new wells vs. prior wells (e.g. (new)/(new + prior)) (if cross-validating)
apri	Temporary adjusted prior production volumes (recalculated in each MC simulation run)
ARIMAfitUpdate	Function for updating ARIMA models
as.year	Function for extracting the year from a calendar date object
binStartStop	Function internal to DCAupdate that finds the start/stop points of decline curve segments
bp	Bar chart object in pregenerated plot for "Field Fractions"
bplotHypDCAcoef	Function for creating boxplots of hyperbolic decline curve coefficients
bplotQfitDCAcoef	Function for creating boxplots of cumulative production curve coefficients
calc_E_dh	Function for calculating emissions from dehydrators
calc_E_fug	Function for calculating emissions from fugitive sources
calc_E_pctrl	Function for calculating emissions from pneumatic controllers
calc_E_ppump	Function for calculating emissions from pneumatic pumps
calc_E_rt	Function for calculating emissions from RICE and turbines
calc_E_sh	Function for calculating emissions from separators and heaters
calc_E_tank	Function for calculating emissions from tanks
calc_E_truck	Function for calculating emissions from truck loading
calc_E_wc	Function for calculating emissions from well completions
calTstep	Function for converting a calendar date object into the equivalent integer MC time step value
cdf.depth.gw	CDF for well depth for gas wells
cdf.depth.ow	CDF for well depth for oil wells
cdf.ff	CDF for field locations / field fractions
cdf.flt	CDF for surface lease owner type by field number

Object	Description
cdf.rework	CDF for well reworks by well type
CDFd	Function for determining the CDF for a vector based on R's density function
cdfDCAcoef	Function for plotting CDFs of decline curve coefficients
CDFq	Function for determining the CDF for a vector based on R's quantile function
clean_names	Function for cleaning up column names that is used internally in the dogmDataUpdate function
clipboard	Function for copying R objects to the user's computer clipboard
DCA.cdf.coef.gas	List object with CDF for each coefficient in the hyperbolic decline curve equation for gas production, organized by field number
DCA.cdf.coef.oil	List object with CDF for each coefficient in the hyperbolic decline curve equation for oil production, organized by field number
DCAlnormFit	Function for log-normal distribution fitting
DCAlnormUpdate	Data analysis function for fitting log-normal distribution to decline curve coefficient trends
DCAupdate	Data analysis function for fitting decline curves to each well in the Uinta Basin
DCAupdateCDF	Data analysis function for creating CDFs for the decline curve coefficients in each field
diffMonPOSIX	Function that returns the number of months between two date objects; used internally in dogmDataUpdate function
dogmDataUpdate	Data analysis function for updating the UDOGM database files used by the rest of the model
Drilled.act	Actual drilling schedule during simulation period. Calculated in postProcess if the model is being run in cross-validation mode.
Drilled.q	Quantile results for drilling schedule
Drilled	MC simulation drilling schedule matrix, with rows for each MC run and columns for each time step
drillingModelUpdate	Data analysis function for fitting distributed-lag energy price models to drilling activity
drillModel	List of fitted drilling models for Equations (3.3) - (3.6)
drillModelData	Data.frame containing all of the data points used in fitting the drilling models
drillsim	MC drilling schedule simulation function

Object	Description
`Ecalc`	MC emissions calculation function
`eci`	List of CPTs for each type of emissions source
`eE`	EBE results list
`Egas`	EIA AEO relative error propagation matrix for gas price forecast. Contains one row for every cumulative probability point in `opt$xq` and one column for every time step in the simulation period.
`EgasFrac`	Same as `Egas`, but for fractional relative error
`eia.hp`	EIA energy price history data.frame
`EIAerrorUpdate`	Data analysis function for determining the direct relative error in EIA forecasts
`EIAerrorUpdateFrac`	Data analysis function for determining the fractional relative error in EIA forecasts
`EIAforecastUpdate`	Data analysis function for converting EIA's AEO forecasts to a monthly basis
`EIApriceUpdate`	Data analysis function for creating the energy price history object `eia.hp`
`EIAsim`	MC simulation function for generating energy price paths via the EIA AEO error propagation method
`emissionUpdate`	Data analysis function for creating the CPT list object `eci`
`emplot`	Post-processing fuction for creating EBE result plots (by species)
`Eoil`	EIA AEO relative error propagation matrix for oil price forecast. Contains one row for every cumulative probability point in `opt$xq` and one column for every time step in the simulation period.
`EoilFrac`	Same as `Eoil`, but for fractional relative error
`eopt`	User input options list for everything related to EBEs
`ep.act`	Actual energy prices during the simulation period. Calculated in `postProcess` if the model is being run in cross-validation mode.
`EPFsim`	Function for controlling which energy price forecast simulation method to use in the MC loop
`eqEcalc`	Function for calculating EBEs in the MC simulation
`eqETsim`	EBE results object created by `eqEcalc` for each loop of the MC simulation
`eqlinecolor`	Line color vector for lines in EIA AEO error CDFs

Object	Description
eqplot	Post-processing fuction for creating EBE result plots (by species and by equipment type)
ETpri	Temporary emissions results list for prior wells (recalculated in each MC simulation run)
ETsim	Temporary emissions results list for new wells (recalculated in each MC simulation run)
fcount	Well counts by field from well.actual data.frame
field	Vector listing each field being analyzed individually and the catch-all Field 999
fitdata	Plane-fit object created to shown LOC regression results in the lease operating cost fits (plot 23)
forlinecolor	EIA forecast line colors
forlinetype	EIA forecast line types
forlinewidth	EIA forecast line widths
gas.q	Quantile results for gas production
gp.ARIMA	List containing the manual and automatic ARIMA fits for gas prices
gp.FC.high	EIA AEO high forecasted gas prices during simulation time period
gp.FC.low	EIA AEO low forecasted gas prices during simulation time period
gp.FC.ref	EIA AEO reference forecasted gas prices during simulation time period
gp.q	Quantile results for gas price paths
gp	MC simulated gas price paths matrix. Contains one row for each MC run and one column for each time step
gr	Temporary gross revenue results for sales of oil and gas for each well (recalculated in each MC simulation run)
gsim	Total gas production for each MC simulation for new wells. Contains one row for each MC run and one column for each time step
GSIM	Temporary gas production matrix used in correcting production volumes based on lease operating costs (see Section 4.5.4, recalculated in each MC simulation run)
hypfit	Function for performing hyperbolic decline curve fitting; internal to DCAupdate function

Object	Description
i	Counter variable used by MC simulation for-loop
ind.wpri	Index of columns which should be selected to pull CPT selections from wpri
ind.wsim	Index of columns which should be selected to pull CPT selections from wsim
ind	Temporary index object, used and overwritten in many different functions
inf_adj	Function for adjusting input terms by an inflation index
j	Counter variable used by postProcess plot counter
leaseCostUpdate	Data analysis function for fitting EIA LOC data
LOC.data	Data used in fitting lease equipment and operating cost models
LOC.gas.equip	Linear regression fit object for capital costs of lease equipment for gas wells
LOC.gas.op	Linear regression fit object for operating costs of lease equipment for gas wells
LOC.oil.equip	Linear regression fit object for capital costs of lease equipment for oil wells
LOC.oil.op	Linear regression fit object for operating costs of lease equipment for oil wells
LOC	Temporary lease operating cost matrix used in correcting production volumes based on lease operating costs (see Section 4.5.4, recalculated in each MC simulation run)
LOCcalc	Function for calculating lease operating costs in MC simulation
mg	DCA results data.frame for gas wells. Limited to the time frame specified by opt$DCA.tstart - opt$DCA.tstop
mgf	DCA results data.frame for gas wells using all available data points
mo	DCA results data.frame for oil wells. Limited to the time frame specified by opt$DCA.tstart - opt$DCA.tstop
mof	DCA results data.frame for oil wells using all available data points
NA.overwrite	Function for replacing NA values with a specified value (by default, 0)
new.p	Actual production of oil and gas from new wells during the simulation period. Calculated in postProcess if model is run in cross-validation mode.
oil.q	Quantile results for oil produced from new wells
op.ARIMA	List containing the manual and automatic ARIMA fits for oil prices

Object	Description
op.FC.high	EIA AEO forecasted high oil prices during simulation time period
op.FC.low	EIA AEO forecasted low oil prices during simulation time period
op.FC.ref	EIA AEO forecasted reference oil prices during simulation time period
op.q	Quantile results for oil price paths
op	MC simulated oil price paths matrix. Contains one row for each MC run and one column for each time step
opt	List object containing all user input options specified in IO_options
osim	Total oil production for each MC simulation for new wells. Contains one row for each MC run and one column for each time step
OSIM	Temporary oil production matrix used in correcting production volumes based on lease operating costs (see Section 4.5.4, recalculated in each MC simulation run)
p	Data.frame containing all of the information in the UDOGM databases proddata, histdata, and welldata for wells located in the Uinta Basin, and just the columns selected in IO_options
path	List of file path directory locations
pb	Counter object used by progress bar that is displayed during the MC simulation loop
pgas.q	Quantile results for total gas production from prior wells
pgsim	Total gas production for each MC simulation for prior wells. Contains one row for each MC run and one column for each time step
pi.skip	Data.frame containing information about skipped prior wells
poil.q	Quantile results for oil produced from prior wells
posim	Total oil production for each MC simulation for prior wells. Contains one row for each MC run and one column for each time step
ppri	Oil and gas production volumes extrapolated from hyperbolic fits of existing wells
prior.Info	Data.frame containing information about prior wells with fits
prior.p	Actual production of oil and gas from existing wells during the simulation period. Calculated in postProcess if model is run in cross-validation mode.
priorInfo	Function for extracting information about existing wells with fits

Object	Description
priorProd	Function for calculating production rates of oil and gas from existing wells with fits
priorProdReworkAdjust	Function for removing production from prior wells that are reworked in each MC simulation run
prob	Data.frame containing cumulative probabilities for well type by field
production	Data.frame containing all of the information in the UDOGM databases proddata, histdata, and welldata for all wells in the state of Utah
productionsim	Function for calculating production rates of oil and gas from new and reworked wells
psim	Temporary production volumes of oil and gas from new wells (recalculated in each MC simulation run)
Q.DCA.cdf.coef.gas	List object with CDF for each coefficient in the cumulative production equation for gas production, organized by field number
Q.DCA.cdf.coef.oil	List object with CDF for each coefficient in the cumulative production equation for oil production, organized by field number
QfitDCAupdateCDF	Data analysis function for creating CDFs of cumulative production curve coefficients by field
qlinecolor	Line color vector for quantile results
qlinetype	Line type vector for quantile results
qlinewidth	Line width vector for quantile results
reE	Reduced EBE results list object
reworkUpdate	Data analysis function for determining cumulative probability of a well being reworked as a function of time
runstart	Time object used to report start time of simulation and calculate the model's total run time
scheduleUpdate	Data analysis function for analyzing many UDOGM database related factors (e.g. actual well data information, well depth CDFs, fields to analyze individually, etc.)
sim_county	Function for picking which county new wells will be located within
sim_DCC	Function used for picking decline curve coefficients for each well in each MC simulation run
sim_depth	Function used for picking well depths for each well in each MC simulation run
sim_dupRework	Function used for duplicating reworked well information in each MC simulation run

Object	Description
sim_EF	Function used for picking emission factors for each well in each MC simulation run
sim_eq_EF	Function for picking rows for CPTs for each well for each type of equipment in the EBE calculation process
sim_fieldnum	Function used for picking field numbers for each well in each MC simulation run
sim_lease	Function used for picking the surface lease owner for each well in each MC simulation run
sim_rework	Function used for picking rework time steps for each well in each MC simulation run
sim_tdelay	Function used for picking time delays for oil and gas production for each well in each MC simulation run
sim_tdrill	Function used for expanding the drilling schedules contained in Drilled into the data.frame wsim
sim_wellType	Function used for picking the well type (oil, gas, or dry) for each well in each MC simulation run
sPR	Simulated production ratio of oil and gas from new wells vs. prior wells (e.g. (new)/(new + prior))
well.actual	Data.frame containing information about all of the wells used in the scheduleUpdate function
wpri	Temporary well information data.frame for prior wells (regenerated in each MC simulation run)
wsim	Temporary well information data.frame for new wells (regenerated in each MC simulation run)