Wilderness Search Planning with rSARP.r

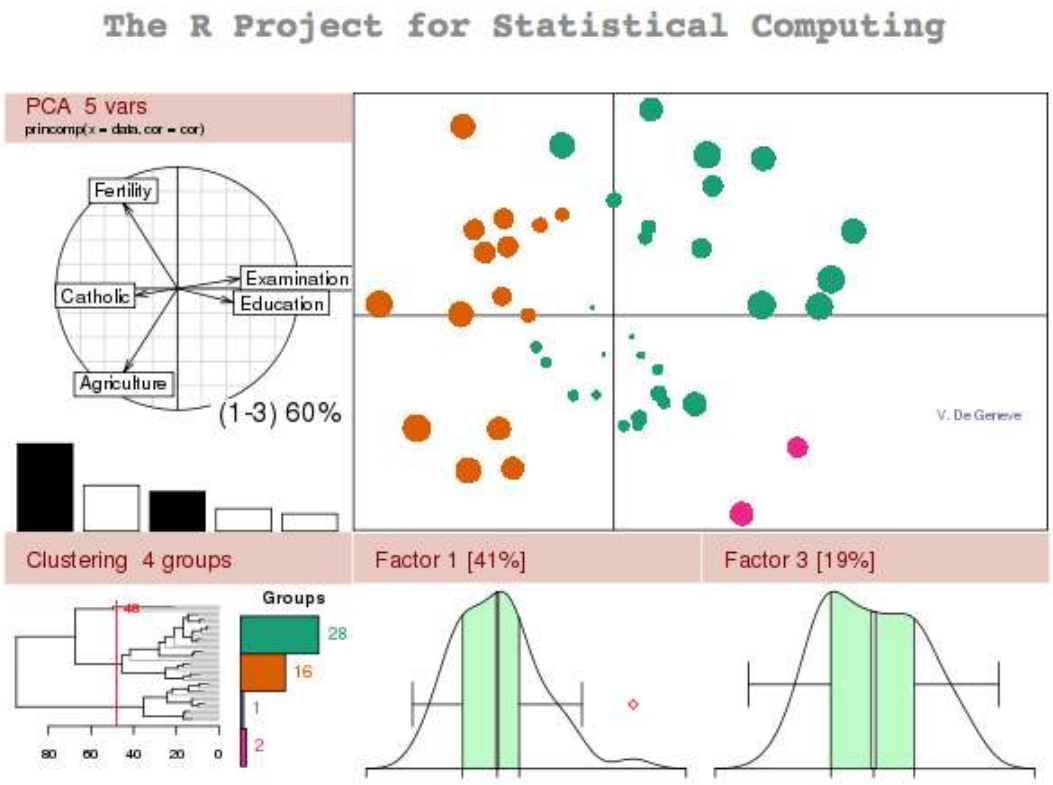
John Hutcheson

2016-03-11

Using R software to effectively and efficiently plan and report wilderness search progress.

Contents

1 Introduction



This report assumes that the user is familiar with Wilderness Search Planning basics, including the complicated process of establishing search zones, Koester Rings, and segmentation (creating search sectors).

1.1 What's this About?

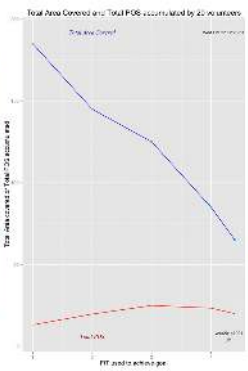
This paper covers the use of the tool set that is available via the rSARP package, which includes the functions searchme(), searchstatus(), tracking(), and bestsearch(). These are programs that a planner can use to craft and report progress against a search plan in wilderness or urban areas to meet the timing needs and resource requirements of a search. The programs can be used to calculate the expected time and resource required to search a given sector (given a limited Michigan based wilderness/urban search model) the expected probability of detection (POD), the probability of success (POS), and the cumulative Overall POS after the search is completed. The program runs in the R programming environment, which is a cross-platform environment with a strong mathematical underpinning. The R system runs and has been tested on a limited basis on Windows and Mac OS X, and provides a method of running extensive math models and producing graphical outputs to help the planner quickly evaluate the most likely outcome of pursuing the preliminary plan, the risks associated with pursuing a given plan, and the best variables to change to improve the plan.

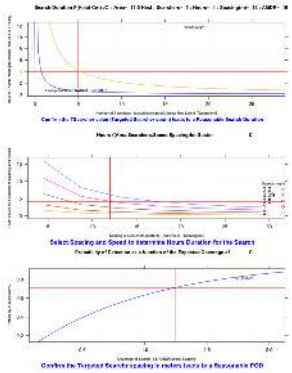
The rSARP package contains mathematical models of the search process that follow the process used by the Midland County Search and Rescue team. It is built on the premises laid out in the National Planning Course provided by the AFRCC and the USCG. The searchme() function is the workhorse of the program, doing most of the heavy lifting. The searchstatus() function is a program that summarizes the current status of a search plan generating easy to interpret graphs, handy for reporting to Search Managers and the general public. The bestsearch() function provides a constrained resource view of the possible ways to deploy a limited number of search resources. And finally, the tracking() function produces a status report of search progress by task (525). This paper documents these programs and their uses.

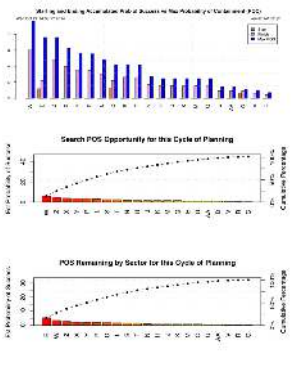
If you find your search plans result in search teams spending more time or less time in the field than is optimal, if you find that search teams are regularly over or under manned, or if you find that search sectors are searched but have to be searched again because sufficient resource wasn't dedicated (low POD), than this program may help you improve performance. If you're spending more time reporting on planning than planning, this program may be of use to you.

But to paraphrase George Box, all models are wrong, and these are no exception. The model used to generate the search estimates are based primarily on searches within Michigan's lower peninsula - which is primarily flat. And the model is based on limited data from a small number of searches. But the framework of the search program is readily adaptable and was designed with scale-ability in mind. Some models are useful - I hope you find that to be the case for these programs.


Alphonse Plan jfh1...


sensitivity1 copy.png


SectorC.png



sectorstatus1.png

2 Preparation

2.1 R Required

The rSARP package cannot be accessed without first installing the R software that allows it to operate. Please see `https://www.r-project.org/` to install R on your operating system. Currently MCSAR planners are using the JGR software layer over R to lower the burden of using R and are considering the use of RStudio. This requires that planners locate the latest version of JGR or RSTudio and R on the web and install these on their computer as a first step. See `http://rforge.net/JGR/` for specific instructions for your operating system to install JGR. See `https://www.rstudio.com/` for specific instructions on how to install RStudio.

Loading the rSARP package into your R application brings in the functions necessary to run the entire model. R has somewhat intelligent loading processes that will warn you if you need to update a package on which rSARP depends or if your R software itself is not recent enough to properly run the software.

- Beware

  The R program will continue using old copies of a package it keeps on hand if you don't correctly install the new version. Fortunately, the version of the rSARP program is reported on the PDF files it creates, so eventually you will discover that your changes didn't take if you fail to properly update the package.

2.2 Create Search Sectors

The searchme() model operates against search sectors which have already been created. These are geographically defined areas usually bounded by track logs that define the areas to be searched. Keep the following in mind as sectors are created:

- Use natural boundaries whenever possible to define sectors. The fewer the natural boundaries available, the more flagging required and the greater the opportunity for navigational errors.
- Limit the size of the search area to match the resources available and target no more than a 6 hour deployment for any team. Any search over 6 hours strains the team and research has shown that search effectiveness drops markedly after 6 hours.
- When in doubt, under-size a sector. Since sectors can be 'ganged' together after prioritization, it's better to pull together little ones than leave a large one unfinished.
- Remember that the strength of the method is in prioritizing the sectors by a group of knowledgeable investigators. The more discrimination in the map in the form of sectors, the easier it will be to attack only the highest priority areas.

2.3 Data Files

The current versions of the functions working within the rSARP package require that a comma spaced value (CSV) data set of a particular layout be installed within the R 'working directory' within your computer. When R is started, a working directory should be identified, then that directory becomes the primary source of input and output files from this package. **A data set, named SearchInput.csv, is the means for providing input data to the program, must be present in the R working directory for the program to function, and cannot be renamed.** Results from running the programs are placed in your working directory or a subdirectory of your choosing and include csv versions as well as graphic files. Please move the supplied data set, SearchInput.csv, into your working directory to run the program.

2.4 Capture Information for the Model

As sectors are created, the information about each sector should be captured and stored by row in the SearchInput.csv file that is stored in your working directory. The first row contains the names of the variables - please don't change the columns A through L in any way, especially the variable names in row 1. The program will crash if it doesn't find the right names in row 1. The program ignores any columns after L. Also, please delete any row that contains incomplete data for these rows (in columns A thru L) before the program is run, or it will crash and complain of missing data. See the example below and the explanation of each column.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Sector	Area	AreaCoverage	Terrain	TOD	WX	THours	TSearchers	TSpacing	AMDR	Rank	POScum
2	A	10	100	HeavyWoods	Day	Normal	-1	-1	-1	-1	313	-5.463
3	B	10	80	SteepTerrain	Day	Normal	-1	-1	-1	-1	151	-2.108
4	C	20	75	OpenWoods	Day	Normal	-1	-1	-1	-1	138	-1.806
5	D	15	50	ModerateTerrain	Day	Normal	-1	-1	-1	-1	151	1.318
6	E	10	25	OpenArea	Day	Normal	-1	-1	-1	-1	275	1.2
7	F	25	100	FlatArea	Day	Normal	-1	-1	-1	-1	112	-1.955
8	G	25	100	Urban	Day	Normal	-1	-1	-1	-1	39	0.681

- Column A: Sector - this constitutes the sector name, typically a letter group though the program will accept any name, and typically will be the letters A-Z, AA-ZZ, AAA-ZZZ. Do not use the letters I or O, since these are often confused with numbers. The name of the sector is printed out on each graphics page as a reference where appropriate.
- Column B: Area - the enclosed area of the sector. The program defaults to area in Hectares (100 x 100 meters), but can also use Acres. The model must use the same area units throughout - you can't mix units within this column(acres, then hectares, then acres). The area unit selected is printed out on the PDF graphical output of the program with the area value.
- Column C: AreaCoverage - The percentage of sector area that is to be or has been searched in this operational period. This should typically be 100%. This column permits calculations to reflect the presence of bogs, lakes, or other areas that were omitted in a search. This column affects most outcomes of the planning process, so a change from 100% in the plan to 8% in the execution should prompt an update of SearchInput.csv and a re-running of searchme() to update the actual POS values achieved in the field.
- Column D: Terrain - the Terrain of the sector that is to be searched. The program currently uses both topographical descriptions (FlatArea or SteepTerrain) and environmental descriptions (OpenArea or HeavyWoods), but not both. A future version may permit both, but for now, it must be one or the other. If you're relying on the model to estimate THours, AMDR and TSpacing values, then please recognize the significance the terrain carries and value it accordingly. Currently only the following values can be used and must be typed exactly as given below:
  - Urban - any area comprised of primarily yards, homes, businesses, parking lots, and groomed parks or golf courses. In essence, if the lawn is mowed, it's Urban. If the grass is "high", it's OpenArea.
  - FlatArea - any area that is flat but may be covered in low brush and relatively Open.
  - OpenArea - any area that is flat but may be covered in low brush
  - ModerateTerrain - any area that is hilly or is not flat but not steep
  - OpenWoods - any area that is relatively flat but wooded.
  - SteepTerrain - any area with steep cliffs or hills that will make travel difficult
  - HeavyWoods - any area which is heavily wooded or full of dense brush
- Column E: TOD - Time of Day. The model essentially doubles all timing estimates as the search plan moves from day to night. Contains one of two values:
  - Day
  - Night
- Column F: WX - Weather. Weather values currently are used to 'shade' search estimates. The use of a WX factor that is different than "Normal" will make small adjustments in the predicted speed of the search, according to the following values.
  - Normal - uses the average predicted speed of the terrain
  - Snow6to8in - subtract .25 mph from the estimated speed of search based on terrain alone
  - VeryHot, VeryCold, Hot, Cold, Raining, Snowing - treated as Normal - no impact.
- Column G: THours - This value is an estimate of the number of hours the team will stay in the field, measured in hours duration of search time (no transport time, break time, or down time is included). Planners should recognize that the target values should generally not exceed 6 hours and that many factors not currently defined or estimated will affect the actual time in the field. Enter a zero (0) for this value if you would like to exclude this sector from actively being searched in this cycle of planning. Enter -1 for this value if you would like the model to estimate this for you. If -1 is entered, the model uses sector Terrain to estimate the searcher speed of travel using the following:
  - HeavyWoods / SteepTerrain = .4 mph
  - OpenWoods / ModerateTerrain = .8 mph
  - OpenArea / FlatArea = 1.2 mph
  - Urban = 1.7 mph
- Column H: TSearchers - The head count of active searchers on the search team. Enter -1 to have the program calculate the optimal team head count. Enter zero (0) to indicate the sector won't receive searchers attention in this cycle of planning. Enter a fixed value if you want all other calculations to use that number of searchers.
- Column I: TSpacing - This is spacing in meters between team members that is planned for the search. This variable affects the Probability of Detection (POD) that is calculated and reported by the model. Enter a negative one (-1) for this value if you would like the model to select a value for you. The model will calculate the widest separation in meters that the searchers can maintain to achieve your POD targets based on the estimated or measured AMDR.

- Column J: AMDR - Average Maximum Detection Range - This is an estimate of the distance measured in a Rain Dance for the object sought. The best plan results from a measured value. The model will estimate a value based on terrain if a negative one (-1) is entered for this value, using the following values:
  - HeavyWoods / SteepTerrain = 4 meters
  - OpenWoods / ModerateTerrain = 7 meters
  - OpenArea / FlatArea = 10 meters
  - Urban = 10 meters
- Column K: Rank - Consensus ranking of each sector from the sector ladder ranking exercise. Enter the total rank figure obtained for each sector (By adding the rank from each person ranking the sector. Use the Score Total column on Form 506).
- Column L: POScum - Cumulative Probability Of Success score - the sum of POS obtained from all previously completed searches in this sector, usually called Overall POS cumulative. Enter zero if the area has not been searched. Copy in the OPOScum value from the previous planning SearchOut.csv sheet if the area was searched as planned and will be searched again. Enter the negative value of the OPOScum column if the sector is no longer considered active.

3 Package Use

The following sections describe how to use the program functions to prepare a practical search plan. The bestsearch() function is usually called first after the sectors are cut and the SearchInput.csv table has been filled out. This function helps management decide whether to search sectors using fast or thorough methods. The searchme() function is typically called next and finalizes the planning with searchstatus() automatically called by searchme() to graphically document the finalized plans. The searchstatus() can be called repeatedly as the SearchInput.csv table is updated to reflect the progress of the search within an operational period.

3.1 Practical Advice

As sectors are created, fill out the SearchInput.csv spreadsheet and save it to your working directory since this file is required by all functions. If this file is somehow lost or misplaced, the output file from the program can be used to generate the same graphs and output. The program ignores all columns after L, but pays attention to any cells below the last row of sectors between A and L. Please don't use cells in these columns below the last sector or the program will not run.

3.2 Calling bestsearch()

Once the data has been stored in the SearchInput.csv file in the proper directory, switch to the JGR program and call bestsearch(). This is done by typing **bestsearch()** in the command line window of the JGR program (the lower window). As soon as the first few characters of bestsearch() are typed, JGR will bring up the program command it believes you want in an italic font complete with any of the input options available between the parentheses it provides. See the example below.

```
bestsearch (AvailablePeeps = 25, sdirectory = "Sensitivity", FTt = c(1,
    2, 3, 4, 5), STitle = "Search Name")
```

All the options available in the program and their default values are presented when bestsearch() is called within JGR. The option name and available states are explained below:

- AvailablePeeps - The number of searchers expected to turn out for this cycle of planning. The bestsearch() function uses this number of searchers to devise and summarize plans for a sensitivity analysis pitting fast search methods against more thorough methods.
- sdirectory - Allows the user to create a new sub-directory in the Searches directory where the detailed search plan graphics will be stored and store all relevant output in the new sub-directory. Defaults to a "Sensitivity" sub-directory. The program will create the sub-directory if it doesn't exist, will warn the user if the directory already exists but will write over it anyway, so please choose directory names carefully.
- FTt - a list of 5 initial FtT values used to perform the sensitivity analysis. It defaults to the integers 1 through 5 but can be set for intermediate options if desired. Please specify this as an R list in the form of FTt = c(1, 2, 3.5, 4.8, 7) if needed. The 5 values need to be placed in order from smallest to largest, and should be greater than zero and less than ten.
- STitle - The formal search name or acronym used for the search name. This will be used to label the graphic output.
- Parameter Use The program allows none, any or all of the options to be revised. To change these values, simply enter the new value preceded by its variable name in any order within the parentheses following bestsearch() or enter new values without names in the exactly the order the parameters are given. Use quotes where indicated by the guide and separate these by a comma if more than one is given. See the examples below. Also be aware that if you misspell the variable names or the restricted values, the program will bomb out. When that happens, simply say my name in vain, correct your previous errors and call the function again.

- Examples of Good Calls:

  bestsearch(25)
  bestsearch(31,"Sensitivity25", STitle= "NMC Smith Search")
  bestsearch(AvailablePeeps=27)
  bestsearch(sdirectory="Sensi",AvailablePeeps=25)
  bestsearch(AvailablePeeps=15,FTt=c(.7,2,3,4.8,6),sdirectory="FirstPass")

- Examples of Bad Calls:

  bestsrch() - Program Name Misspelled
  bestsearch((- Unbalanced Parenthesis
  bestsearch(sdirectry = SearchTest1) - sdirectory incorrectly spelled and no quotes around the SearchTest1 variable

3.3 Calling searchme()

Once the data has been stored in the SearchInput.csv file in the proper directory, switch to the JGR program and call searchme(). This is done by typing **searchme()** in the command line window of the JGR program (the lower window). As soon as the first few characters of searchme() are typed, JGR will bring up the program command it believes you want in an italic font complete with any of the input options available between the parentheses it provides. See the example below.

```
searchme (FtT = 3, filout = "SearchOut.csv", AreaUnits = "Hectares",
    graphs = 1, directory = "Searches", STitle = "")
```

All the options available in the program and their default values are presented when searchme() is called within JGR. The option name and available states are explained below:

- Parameter Use

  The program allows none, any or all of the options to be revised. To change these values, simply enter the new value preceded by its variable name in any order within the parentheses following searchme or enter new values without names in the exactly the order the parameters are given. Use quotes where indicated by the guide and separate these by a comma if more than one is given. See the examples below. Also be aware that if you misspell the variable names or the restricted values, the program will bomb out. When that happens, simply say my name in vane and call the function again and correct your previous errors in spelling.

- filout - The name of the output file created when the model has completed calculations. Defaults to "SearchOut.csv" as the name of the output file. To create a different file name, enclose the alternate name in quotes (single or double) and stay within the limits imposed by your operating system for file nomenclature. See the examples below.

- AreaUnits - The units you choose to use to report the area of the sector you are planning to search. Defaults to "Hectares". User may enter "Acres" as the alternate area unit - no other alternate area units are currently accepted. See the example below.

- FtT - Fast to Thorough factor. Choose between Fast Search (values less than 3) or Thorough Search methods (values more than 3) by adjusting this value. Values between 0 and 10 are accepted. Defaults to 3 (63% POD), which is usually the optimal balance between POD and resource.

- graphs - Option to control whether sector graphs are created or not. Does not affect the output of the summary status graph (barchart and paretos of all the sectors). This permits the planner to rapidly adjust the plan in evaluation mode, without documenting each sector. Defaults to 1 - Producing graphs. Any value other than 1 (0 or 2 for instance) will result in skipping the graphic output by sector giving nearly instantaneous results.

- directory - Allows the user to create a new sub-directory in the working directory and store all relevant output in the new directory, but defaults to the working directory. The program will warn the user if the sub-directory already exists but will write over it anyway, so please choose directory names carefully.

- STitle - The formal search name or acronym used for the search name. This will be used to label the graphic output.

- Examples of Good Calls:

  searchme(filout = 'Dirks Revenge Sample Output.csv')
  searchme(AreaUnits = 'Acres', filout='Test1.csv',STitle="BCB Search")
  searchme(AreaUnits = 'Acres', filout='Joes Folly.csv', FtT=2.5)
  searchme(filout = 'Battershells bomb.csv', AreaUnits = 'Hectares', graphs=3, directory='Search53B')
  searchme()

- Examples of Bad Calls:

  search() - Program Name Misspelled
  searchme((- Unbalanced Parenthesis
  searchme(fileout = SearchTest1.csv) - filout incorrectly spelled and no quotes around the SearchTest1.csv variable

3.4 Calling searchstatus()

Once the data has been stored in the SearchOut.csv file in the proper directory, switch to the JGR program and call searchstatus(). This is done auto-magically when searchme() is run but can be called outside of that function. This is done by typing **searchstatus()** in the command line window of the JGR program (the lower window). As soon as the first few characters of searchstatus() are typed, JGR will bring up the program command it believes you want in an italic font complete with any of the input options available between the parentheses it provides. See the example below.

```
searchstatus (filout = "SearchOut.csv", directory = "Searches", STitle)
```

All the options available in the program and their default values are presented when searchstatus() is called within JGR. The option name and available states are explained below:

- Parameter Use

  The program allows none, any or all of the options to be revised. To change these values, simply enter the new value preceded by its variable name in any order within the parentheses following searchstatus or enter new values without names in the exactly the order the parameters are given. Use quotes where indicated by the guide and separate these by a comma if more than one is given. See the examples below. Also be aware that if you misspell the variable names or the restricted values, the program will bomb out. When that happens, simply say my name in vane and call the function again and correct your previous errors in spelling.

- filout - The name of the output file created when the model has completed calculations. Defaults to "SearchOut.csv" as the name of the output file. To create a different file name, enclose the alternate name in quotes (single or double) and stay within the limits imposed by your operating system for file nomenclature. See the example below.

- directory - Allows the user to create a new sub-directory in the Searches directory and store all relevant output in the new directory, but defaults to the "Searches" directory. The program will warn the user if the sub-directory already exists but will write over it anyway, so please choose directory names carefully.

- STitle - The formal search name or acronym used for the search name. This will be used to label the graphic output.

- Examples of Good Calls:

  searchstatus('Dirks Revenge Sample Output.csv')
  searchstatus(filout = 'Battershells bomb.csv', directory='Search53B')
  searchstatus()

- Examples of Bad Calls:

  search() - Program Name Misspelled
  searchstatus((- Unbalanced Parenthesis
  searchstatus(fileout = SearchTest1.csv) - filout incorrectly spelled and no quotes around the SearchTest1.csv variable

3.5 Calling tracking()

Once the 525 tracking data has been stored in the tracking.csv file in the working directory, switch to the JGR program and call tracking(). This is done by typing **tracking()** in the command line window of the JGR program (the lower window). As soon as the first few characters of tracking() are typed, JGR will bring up the program command it believes you want in an italic font complete with any of the input options available. See the example below.

```
tracking (title = "MCSAR Search", directory = "Searches")
```

All the options available in the program and their default values are presented when tracking() is called within JGR. The option name and available states are explained below:

- title - The name of the search displayed in the title of the tracking graph. In the example given "MCSAR Search" (the default value) was used.

- directory - Allows the user to create a new sub-directory in the working directory and store all relevant output in the new directory, but defaults to the "Searches" directory. The program will warn the user if the sub-directory already exists but will write over it anyway, so please choose directory names carefully.

4 Search Model

4.1 Model Use In Search Planning

The search model contained within this package is used to rapidly plan, critique, and finalize a search plan and has been tested and used extensively in Michigan in wilderness searches. The program, other than for tracking, is of limited value until the initial search efforts are well under way, search sectors have been established, and the sector ladder has been finished. The program consists of 3 primary functions which are used within an operational period to manage and document search

plans and search progress. The searchme() model does not address or include hasty or initial search efforts or plans, though the tracking() function does.

Once sectors have been created and ranked, the bestsearch() function is typically the first function called to establish the best general approach to searching sectors after the initial (hasty) search efforts are completed or there is available resource to begin loose or tight grid search efforts. This function helps identify the area that can be effectively searched with a limited number of resources as a function of the searcher spacing relative to the sector AMDR. The function understands the relative priority of the sector areas as described in the search table (SearchInput.csv) and aligns the area priorities with the resources available.

After the method of search (rapid, thorough, or in-between) is selected, the search plan is developed using the searchme() function by specifying the relative speed of search (using the FtT parameter) and fine tuning any resource and timing assignments the planner feels are appropriate. Additionally, areas that are planned to be searched within the operational period can be clearly identified as can the areas that will not be searched, either because management considers these areas completed (inactive) or because sufficient resource is not available to consider searching those areas within the period.

The searchme() function can be used in two primary modes as concerns detailed sector planning. In ”manual mode” the planner sets targets based on his experience and uses the software to document his selections. In ”auto mode” the planner relies on the program to directly calculate the expected outcomes using the estimates which hinge primarily on the Terrain values. The function is flexible and can estimate all or none of the THours, TSearchers, TSpacing, or AMDR values at the user’s discretion.

For each sector, the model produces 1.) tabular output in the form of a CSV file describing all sectors and 2.) a single page for each sector containing 3 graphs that display the 3 primary relationships that concern planners when developing sector search plans. The graphs of each sector push the choices made against the Rules of Thumb and NASAR Physical models allowing for a sensitivity analysis by sector. In addition, the console output from the program contains sorted versions of the search plan table which can be used to quickly review and modify plans. See the example below.

```
> searchme()
[1] "FtT for this run was:  3"
[1] "Program completed successfully.  Please check the following location for output/original files:  /Users/NX8A/Dropbox/SAR Planning Tools/Searches"
   Sector Area AreaCoverage         Terrain TOD     WX THours TSearchers TSpacing AMDR Rank POScum fnCovered fnMissed AreaCov   POC terspd terheads FirstPass ROThrs
21      W   25          100           Urban Day Normal   3.05          2     15.0   10  350  0.000      1.00     0.00    25.0 9.666    1.7   6.0923         1  6.178
24      Z   20          100           Urban Day Normal   2.44          2     15.0   10  275  0.000      1.00     0.00    20.0 7.595    1.7   4.8738         1  4.942
22      X   20          100      HeavyWoods Day Normal   5.75          9      6.0    4  226  0.000      1.00     0.00    20.0 6.241    0.4  51.7842         1 19.768
23      Y   20          100        OpenArea Day Normal   3.45          2     15.0   10  202  0.000      1.00     0.00    20.0 5.579    1.2   6.9046         1  4.942
14      P   25          100           Urban Day Normal   3.05          2     15.0   10  201  0.000      1.00     0.00    25.0 5.551    1.7   6.0923         1  6.178
```

4.2 Signi**fi**cant Relationships

The following list maps out the significant relationships between the input variables and the model outcomes within the searchme() model.

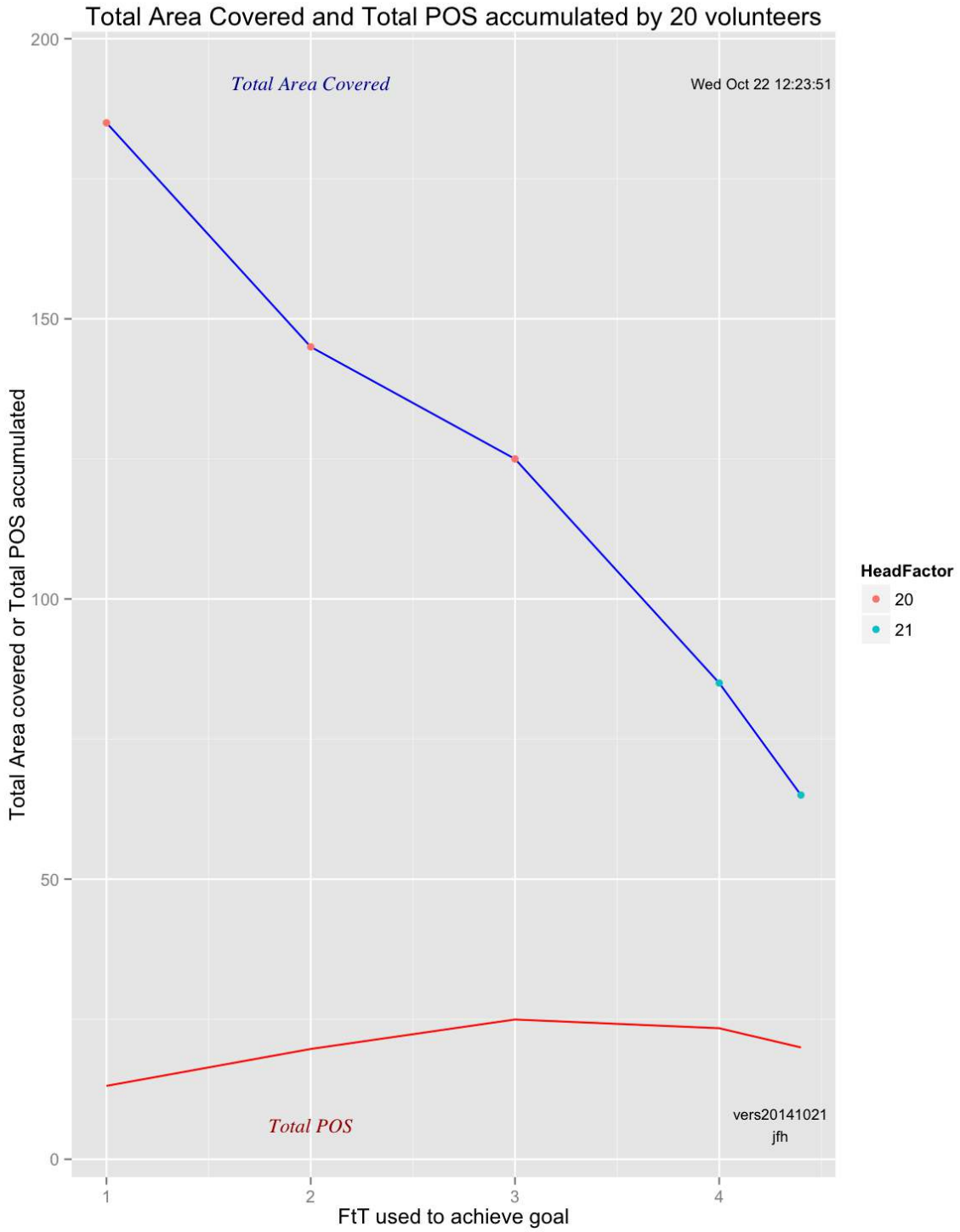
- Column A: No Affect on any outcomes but used extensively in all reporting of sector names.
- Column B: Area affects all outcomes and is proportional to THours, TSpacing, Searchers, ROThrs, NHRSH, terheads, and SAVHRS.
- Column C: AreaCoverage affects all outcomes. Same as Column B. **Set this value to a negative value if the sector was partially searched in a previous period and this portion of the sector is being searched for the first time.**
- Column D: Terrain can be used by the model to estimate the speed of searchers and the AMDR of the area searched. The speed of searchers can be used to control the number of searchers estimated as well as the time to complete the task. Terrain isn’t used to estimate speed unless the user enters -1 for THours, TSearchers, or both. If not used, the model frames the user’s estimates graphically for reference to the Terrain based and NASAR models. If the user enters -1 for THours or TSearchers, the model sets the speed of the searcher by the terrain. If the user enters -1 for AMDR, the model uses Terrain to estimate the AMDR. If the user enters -1 for TSpacing, the model sets the spacing on the AMDR.
- Column E: Time of Day doubles the expected duration of a search for night time searches. The Rule of Thumb includes the TOD affect and is automatically doubled for night time searchers. NASARs estimates don’t include this affect since it’s a “one speed fits all” model. So expect to see the ROThrs column keep pace with TOD with no change to NHRSH column. Graphically the blue line will markedly change with TOD while the brown NASAR line will be fixed. The current model Time of Day has no affect on estimated AMDRs - these must be measure in the field to be reliable.
- Column F: WX values affect the terrain speed estimates by adding or subtracting from the average terrain speed. The current model has very limited WX conditions impacting the outcomes of the model due to the limited data available.
- Column G: THours values are calculated from all model variables if the user enters -1 and are used to frame the plan graphically if values are entered.
- Column H: TSearchers values affect the number of elapsed hours required to search the area. A sensitivity analysis around the planned value for this variable is used to estimate the SAVHRS value, the dPOD and dPOS values.
- Column I: TSpacing values affect the POD, POS, dPOD and dPOS values and the THours to search the area.
- Column J: AMDR values affect the POD, POS, dPOD and dPOS values of the plan.
- Column K: Rank values are used to calculate the POC, POS, and OPOScum values.
- Column L: POScum values are used to calculate the OPOScum values, and if zero indicate that no previous searching in the sector has been done. **If the value is negative, it indicates the sector is no longer active and documents the final OPOScum value obtained in the sector.**

5 bestsearch() Use

The bestsearch() function works by repeatedly calling the searchme() function across various Fast to Thorough (FtT) parameters to identify the optimal search conditions and saves all output from this work in the directory named at the time of the function call by the sdirectory parameter. The output includes all output tables that the searchme() function normally produces, the summary graphs from searchme(), and final summary graphs of the bestsearch() function. The bestsearch() summary graphs are given by example below.

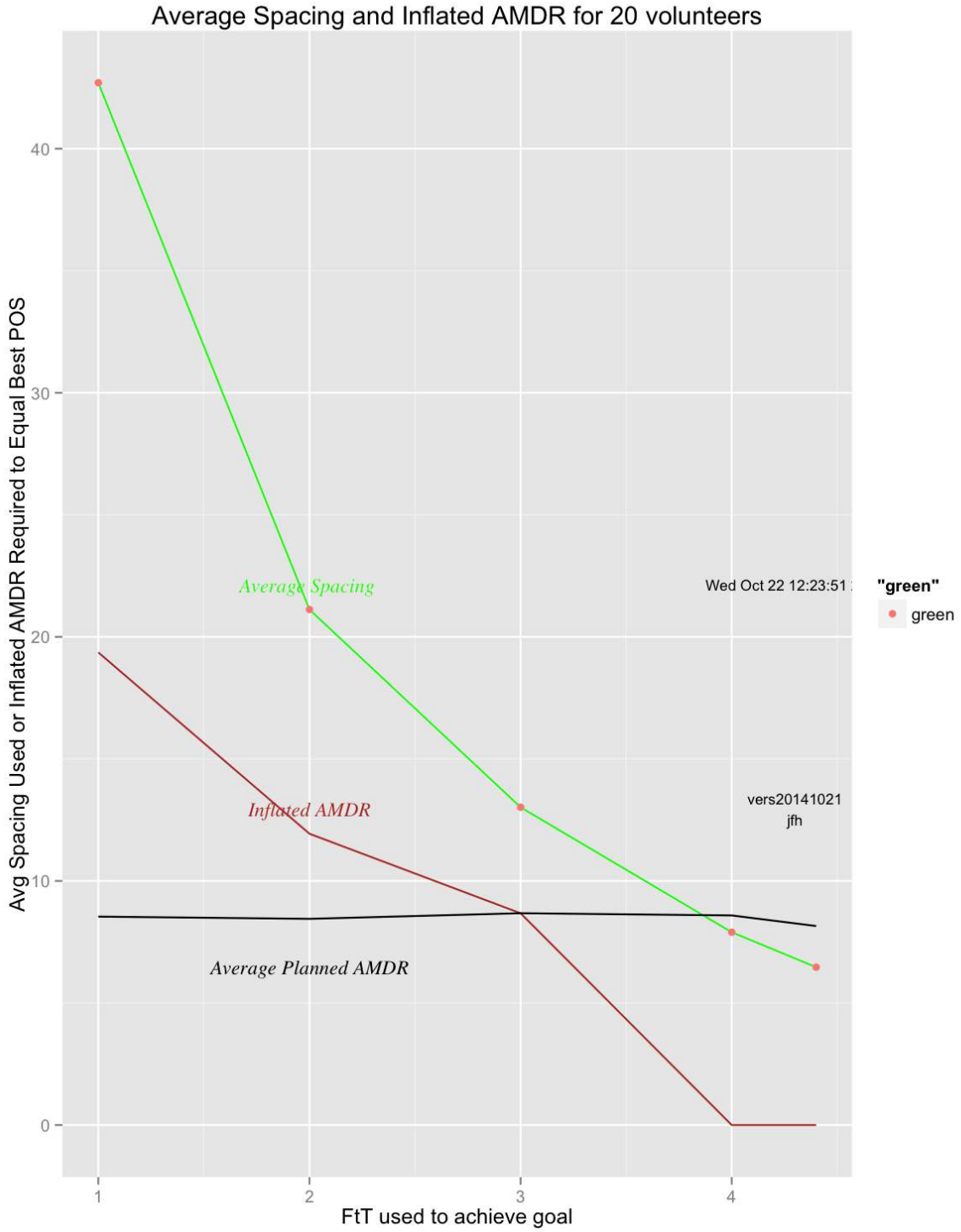
The first page examines the area that can be covered by the resource available and the expected POS across this area as a function of the FtT parameter. The FtT is a parametric form of the probability of detection (POD).

Since bestsearch() is attempting to find solutions with so many constraints, sometimes the program finds no solutions. The function first varies the number of searchers up and down by one person, and then varies the FtT down by .05 to find a solution. It continues to vary these parameters until it either finds a solution or reaches an FtT which is less than .6 of the original target. If no solution is found, it reports all values as zeroes.



The second graph given by the bestsearch() function examines the average AMDR used in the search plan, the average spacing between team members, and a parameter called Inflated AMDR. This represents an estimate of the value the average

AMDR would have to inflate to in order to achieve the highest reported POS (shown on the previous page) when using rapid search methods. Since AMDR figures are typically based on visual search methods and do not credit responsive or mobile subjects or audio calls, this is one way of answering the question how much risk is borne by using the faster methods or how much does the average AMDR have to change to make the faster methods equal in attractiveness to the more thorough methods.



The inflated AMDR curve crosses over the spacing curve at the method that had the highest POS and is zeroed out for the higher levels of search POD to avoid confusion.

I recommend planners use these curves to discuss with search management the likelihood the subject can be found using more rapid search methods and in determining which search method is most appropriate.

6 searchme() Use

6.1 Basic Operation

The searchme() program first creates a set of tables in the console that can be inspected and used by the planner to rapidly modify the plan without relying on the graphical output.

The first output table produced is sorted by POS to highlight the hightest value sectors first. This is followed by summary lines of text describing the total resources required for this search plan. Then the same tables are reprinted to the console, but in alphabetical order by sector.

6.2 Tabular Output

In addition to all the input columns saved, the program output includes the following columns:

- fnCovered - the fraction of area covered by the search plan or actual search results.
- fnMissed - the fraction of the area missed by the search plan or actual search results.
- AreaCov - the area in hectares or acres that was covered in the search.
- POC - Probability of Containment for the sector. A normalization of the sector ladder totals, this is the ranking of the sector over the other rankings. An increasing Sort of sectors by this column would place the most important sectors at the top and the sectors least likely to contain the subject at the bottom. Formerly known as Probability of Area (POA).
- terspd - Terrain Speed in miles per hour. This is an estimate of the speed of progress of the search using the terrain to estimate the speed based on previous MCSAR experience in Midland County. IF the program is given the headcount and time planned for the search by the planner, this number is calculated and not estimated by terrain.
- terheads - Terrain Man Hours. This is either an estimate of the total man hours required to search the area using a geometric model and the terrain to estimate the speed of progress, or if the planner has fixed the TSearcher headcount and the THours time in the field, is a calculated value of the planned hours in the field. If the value is estimated by the model, it is based on the terspd, area, TSearcher and TSpacing, and WX values and is the most reasonable estimate we currently have available for searches with no additional information. See terspd for the speed.
- FirstPass - value that reflects if sector should skip Bayesian treatment (-1) and be treated as if the area is searched for the first time.
- ROTHours - this is the estimate of the hours to search based on data given in Kent County's Search Management course in 2007/2008. Our current understanding of it is that it's an optimistic estimate of timing for an Urban Search.
- NHRSH - This is an estimate of the total man hours required to search the area using a geometric model and the NASAR assumption of .29 mph speed (pucker-brush). This represents the brown line on the graphical output and is a very conservative estimate of the time to search during the day. For night time search, it is often over ranged.
- SAVHRS = Saved Hours. This is the time saved if an additional person is added to the team (TSearchers increments One) and the spacing between members is kept constant, increasing the speed of search without affecting the thoroughness (POD) or probability of success (POS). The sum of this column is the time savings expected in completing the search if one person were added to each team and the sectors are sequentially searched.
- ESW - Effective Sweep Width in meters. Its assumed to have a value of 1.5 * AMDR based on the Koester et al 2004.
- COV - Coverage. The ratio of the Effective Sweep Width and the actual spacing used in meters [(ESW) / TSpacing]. This number is used to calculate POD and is shown on the X axis of the POD graph (see below).
- POD - Probability of Detection for the sector based on TSpacing and AMDR. An estimate of the thoroughness of the search. Displayed on the POD graph.
- delspc - the change in team spacing the addition of a single person to the team would have if the addition was added in such a way as to maintain the line overall line length.
- dPOD - the change in the POD the addition of a single person would have if the spacing was allowed to decrease by delspc. This estimate is similar to the SAVHRS estimate, but with the emphasis on increasing thoroughness of the search.
- POS - Probability of Success for the sector. An estimate of the odds of finding the subject in the sector if plans are executed flawlessly and estimates are accurate. The result of multiplying POC x POD expressed as a percentage.
- dPOS - delta Probability of Success. An estimate of the impact dPOD would have on POS if headcount on the team were allowed to increase by one as described in dPOD. The sum of dPOS column is the increase in the Probability of Success expected if one person were added to each team and the addition allowed to increase the thoroughness of the search.
- OPOScum - Overall POS cumulative - the running final POS for a sector including all POS value obtained from previous searches. Assumes the plan is executed as planned. This value is copied into the POScum column when a new plan is drafted.
- RTotlHeads - Running total count of the searchers used by row. Does not include Field Team Leaders if they do not add to the search line.

- RTotlPOS - Running Total POS - cumulative POS by row.
- TotlArea - Running Total Area - cumulative area covered by row.
- AvgSpacing - Running Average Spacing - average for the row and all previous rows.
- rPOC - Running POC - cumulative POC for the area covered by row for this operational period. Does not include POC already claimed nor POC not covered due to any area missed.
- AvgAMDR - Average running AMDR - average AMDR for this row and all previous rows.
- POSm - POS Missing - the POS that was lost due to the area that was missed or hasn't been searched (due to fnMissing). This column does not count missed POS if the sectors are moved to the "complete" status by setting the POScum value to a negative value.

6.3 Searchme() text output

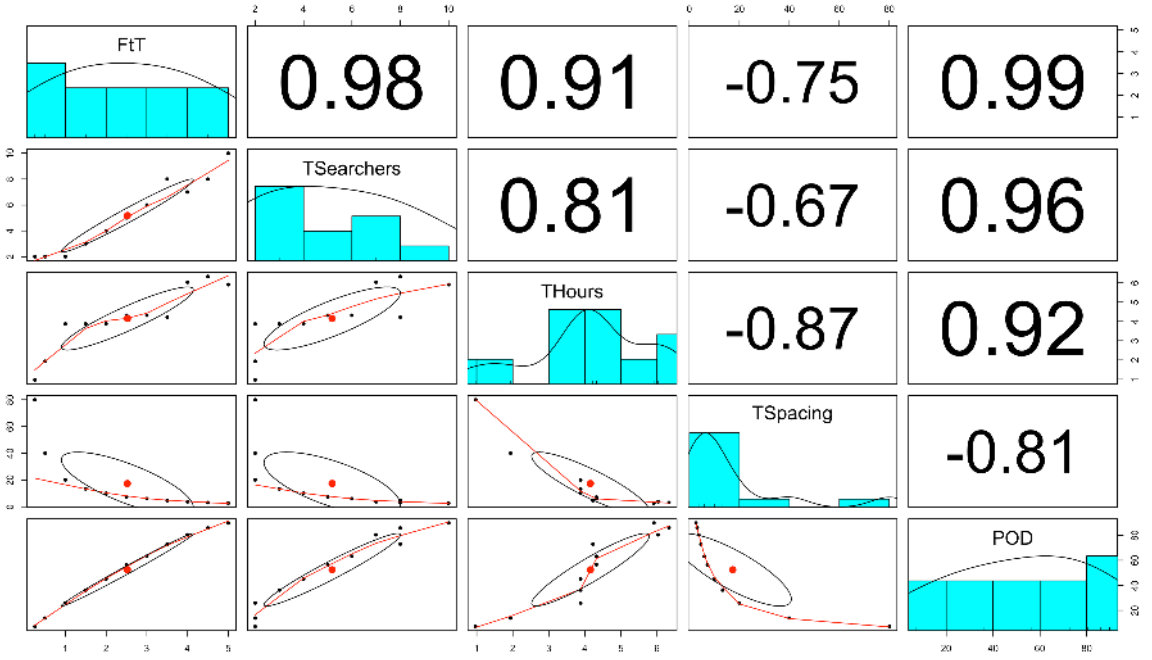
The searchme() function provides a text summary of the resulting search plan for users. Example output is given below:

- Total Probability of Success if all sectors searched as planned and all sectors rated correctly = 63%.
- Impact of missing areas in LOST POS is 3.5%
- Total Searchers Required if all sectors searched as planned = 120 peeps
- Total elapsed Hours to execute this plan if all sectors searched in series = 75.1 Hours
- Total Manhours consumed if all sectors searched as planned = 410.48
- Additional hours saved in search if 1 person added to each team and spacing held constant= 14.25135 hrs
- Increased POS if 1 person added to each team and spacing decreased accordingly= 6.447956 %
- First Sector to consider adding in more peeps to improve POS is Sector G
- First Sector to consider adding in more peeps to shorten the Search hours is Sector D
- Status Report successfully completed.

6.4 Fast vs Thorough Planning

The searchme() program allows the planner to adjust the plan across all sectors which are controlled by the model from ridiculously fast to ridiculously thorough. The FtT value given by the user will automatically shift from a default target of 63.2% POD to whatever relative value the planner chooses to hit. Adjusting the value adjusts the spacing between searchers, which in turn increases or decreases the other calculated values in the program, and permits the planner to rapidly evaluate a number of planning approaches.

To help planners understand the complexity of the relationships between the program variables, see the graphic summary of the output values from the model when the FtT value is adjusted for a fixed area with a fixed Terrain. The large black numbers are the Correlation coefficients between the named variables in the row/column the number is shown at the intersection. The blue charts are bar charts showing the range of values of the variable, and the other diagrams are dot plots showing regressions between the variables.

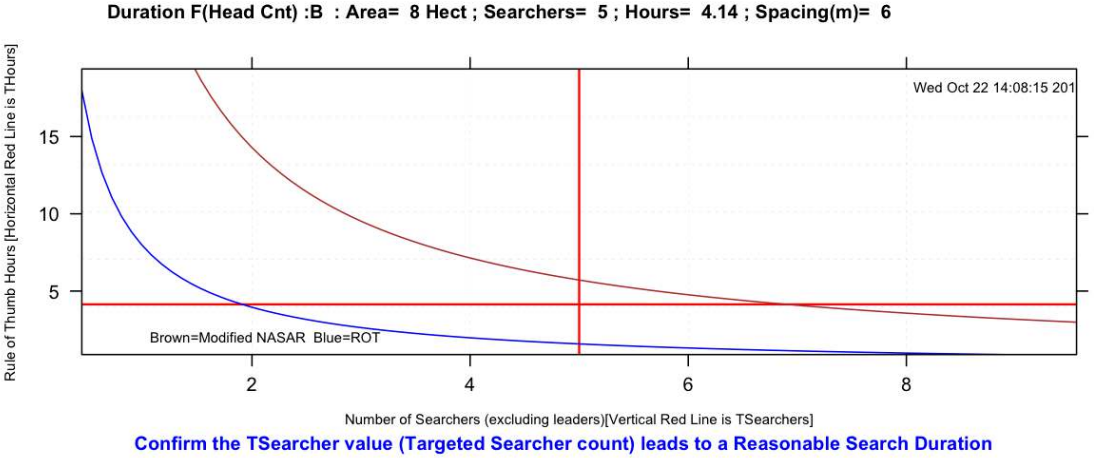


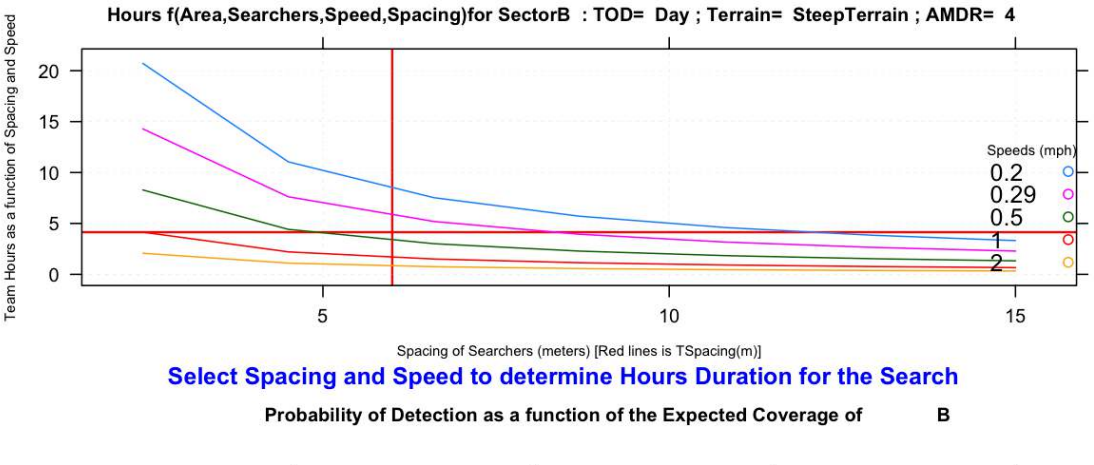
6.5 Graphic Output

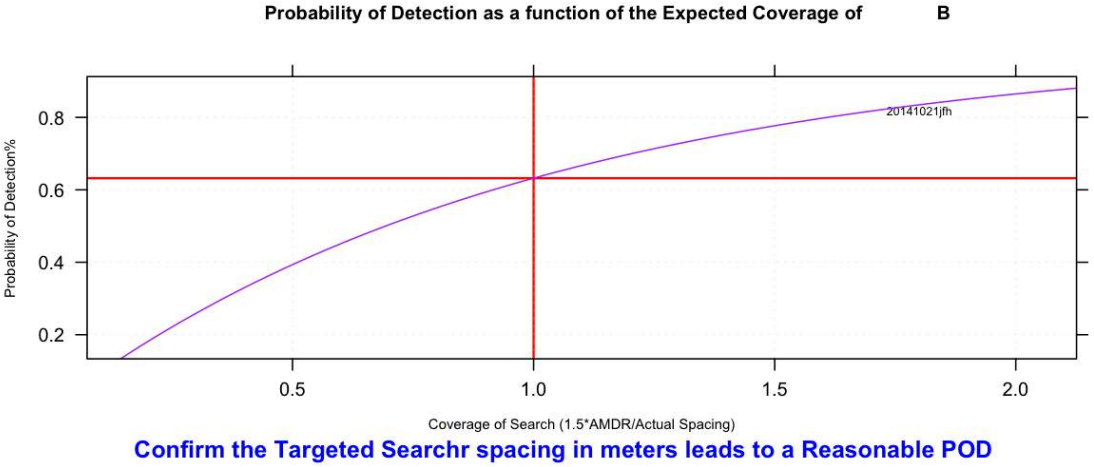
The searchme() function defaults to create graphs for each sector allowing planners to do a quick sensitivity analysis. Due to the continued improvement of the search model, these are seldom used but still available for use. Please see the example and the explanations below for the use of each graph.

- First Graph - Searcher Count vs Search Hours

  The graphics page displayed below is an example of the output from searchme(). The first chart on the page is a graph of the relationship between the expected hours the team will spend in the field on the y axis and the number of searchers placed in the field. The Brown line is a depiction of the "worst case estimate" that NASAR recommends - using a speed of .29 mph. While MCSAR has seen 1 or 2 searches that required more time, this is a reasonable worst case estimate and should be considered as such by planners. The blue plot line is an estimate of the 'Rule of Thumb' presented at Kent County's search management training in 2007/2008. It represents a best case example and has only been equaled in Urban searches. The horizontal red line on the graph represents the number of hours expected or planned for this sector (THours value), and the vertical red line represents the number of searchers planned for the sector (TSearchers value). The intersection of the two red lines is where the planned search falls relative to the worst and best case outcomes.







As the graph indicates, the user should examine the intersection of the red lines and evaluate the risks associated with the plan by inspecting the outcomes. The number of searchers (TSearchers value) can be adjusted, and the graph can

be used to estimate the impact this would have on THours by intersecting the valid search space (the area between the brown and blue lines) at the same relative location as the original intersection. In the example above, a shift from 5 searchers to 4 searchers would lengthen the search to approximately 6 hours from 4 hours. Moving from 5 to 8 searchers would shorten the search to about 3 hours. Likewise, by inspection dropping the team to even 3 members risks the search taking more than 6 hours - an unwise plan. The first graph is used to confirm that the number of searchers planned for the team will likely result in the team completing it's task in a reasonable time. If the first graph isn't satisfactory, don't bother to inspect the others. They'll be unsatisfactory as well.

- Second Graph - Speed and Spacing vs Search Hours

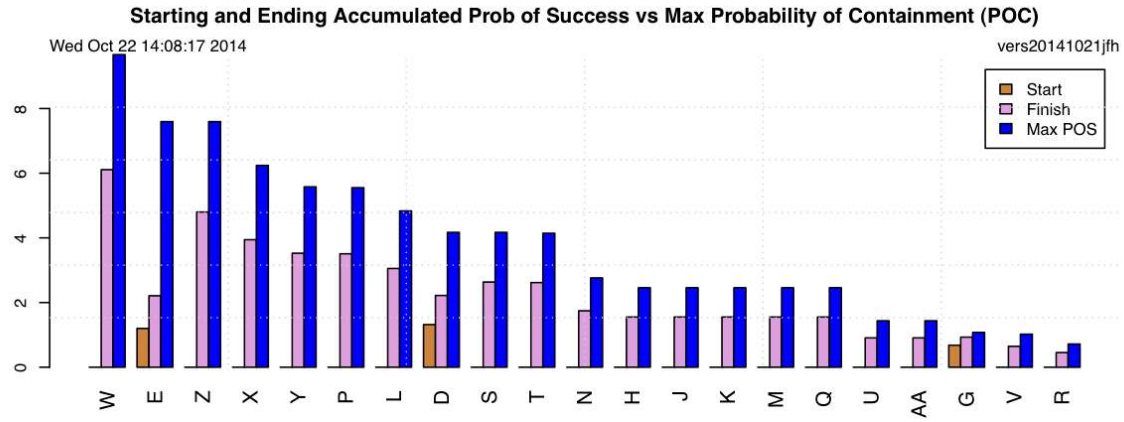
  The second chart on the page is a graph of the relationship between the expected hours the team will spend in the field on the y axis and the spacing between searchers and the speed they will need to conduct the search in the time allotted. The colored lines in the graph depict a range of speeds that are reasonable for teams to target for ground search. The blue depicts the worst case observed speed of .2 mph seen in a MCSAR search. The pink line is a depiction of the "worst case estimate" that NASAR recommends as a default speed of .29 mph. The orange speed at the bottom represents 2 mph and should represent a top speed for most searches. The horizontal red line on the graph represents the number of hours expected or planned for this sector (THours value), and the vertical red line represents the spacing of searchers planned for the sector (TSpacing value). The intersection of the two red lines is where the planned search falls relative to the speed of the search. As the graph indicates, the user should examine the intersection of the red lines and evaluate the risks associated with the plan by inspecting the outcomes. If the planned spacing of the searchers results in a reasonable speed, all is well. If the speed is too fast, either the spacing will have to be increased, or the number of searchers increased. In the example above, you can see that as the spacing drops to 5 meters, the time to complete the search at the same speed rises to approximately 8 hours. To keep to the 4 hour search, the searchers have to approach .5 mph. If the spacing drops below 5 meters, it's doubtful that the search team could maintain that speed. The second graph is used to confirm that the speed of the searchers planned for the team and the team spacing will likely result in the team completing it's task in a reasonable time.

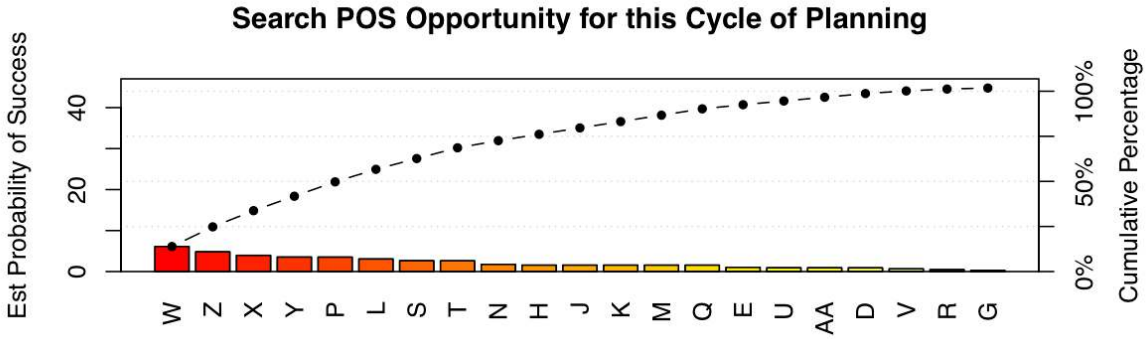
- Third Graph - Spacing vs Probability of Detection

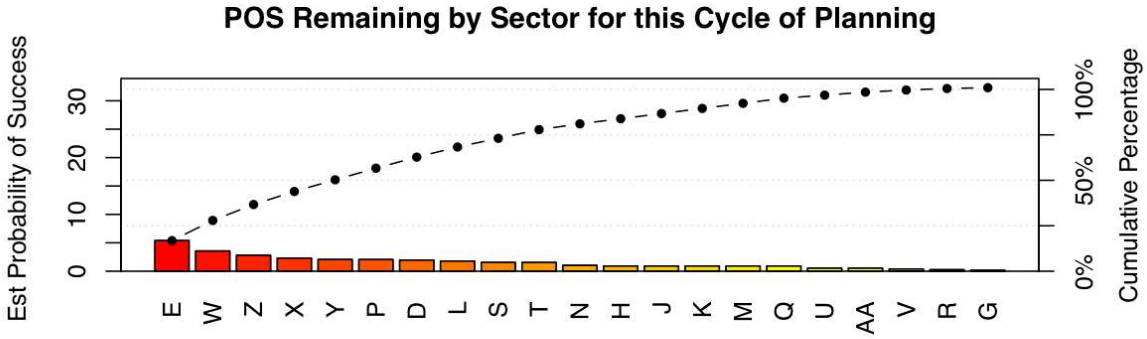
  The third chart on the page is a graph of the relationship between the expected team spacing on the x axis and the Probability of Detection on the y axis. The purple line in the graph depicts the relationship between the COV (coverage) and the POD. COV is inversely proportional to TSpacing. The horizontal red line is the expected POD of the sector. The vertical red line is the COV resulting from the TSpacing value. As the graph indicates, the user should examine the intersection of the red lines and insure that the TSpacing gives a reasonable POD. The general consensus in Michigan is that sectors searched so that resulting PODs fall below 63% have not been adequately searched and will need for subsequent follow up searches. If the third graph isn't satisfactory, adjust the TSpacing or the TSearchers or the THours values and run the model again.

7 searchstatus() Use

The searchstatus() function works by examining the output file from the searchme() function. If users change the default name of this file when creating it in the searchme() function, they must properly name it when calling the file in the searchstatus(). The output includes all output tables that the searchme() function normally produces and final summary graphs of the searchstatus() function. The searchstatus() summary graphs are given by example below and include three pages that examine the sectors that are yet to be searched (active) and one page that examines the pages that are listed as completed (inactive).







Example First, Second, and Third Graphs for Active Sectors

7.1 Active Sectors

For sectors which are still considered to be viable for future search activity, the following graphs are produced. Active sectors have a positive or zero POScum value (column L).

- First Graph - Start, Finish, and Maximum Probability of Containment

  The first graph is a bar chart summary of the POC/POScum values for each sector, sorted in decreasing order of POC. The start value is the value given for the POScum listed in the output file. This represents the amount of POS that has already been 'harvested' in earlier searches before the current plan is implemented. The finish value is the OPOScum value and represents the amount of total POS 'harvested' after the current plan is implemented. The max POS is the most POS that can be obtained from the sector and is the value of the POC. POS is the probability of finding the subject in the sector given the resources available and the probability of detection (POD). This bar chart then summarizes the

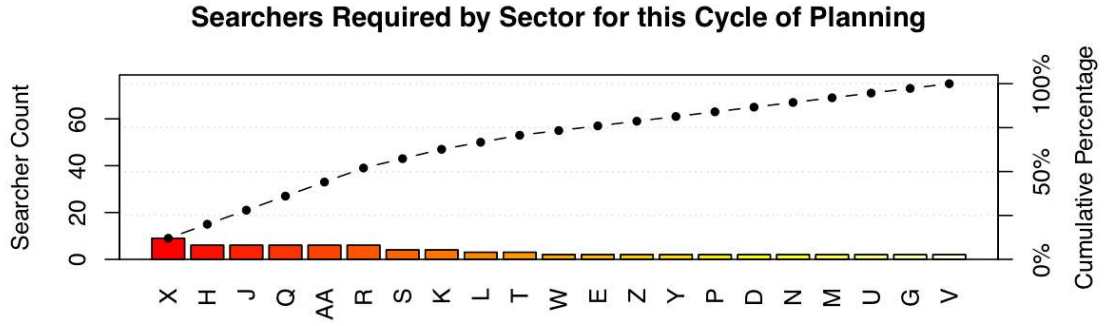
plan (from start to finish) in a single view. It does not list any sectors that have been completed - these show on page 4 (if any).

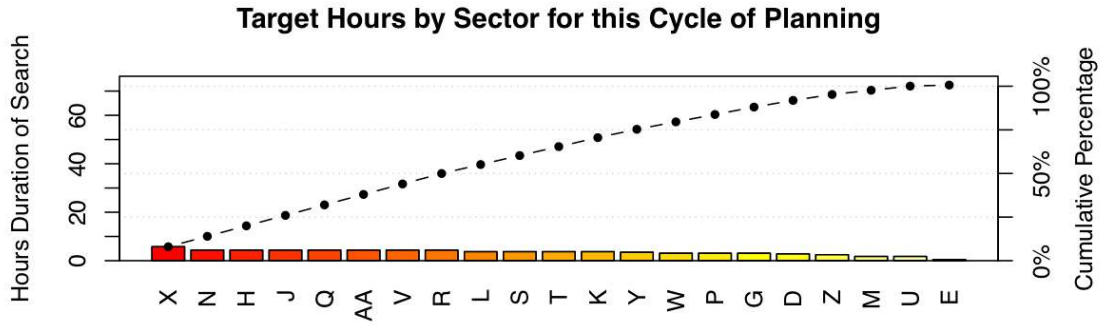
- Second Graph - Search POS Opportunity for this Cycle of Planning

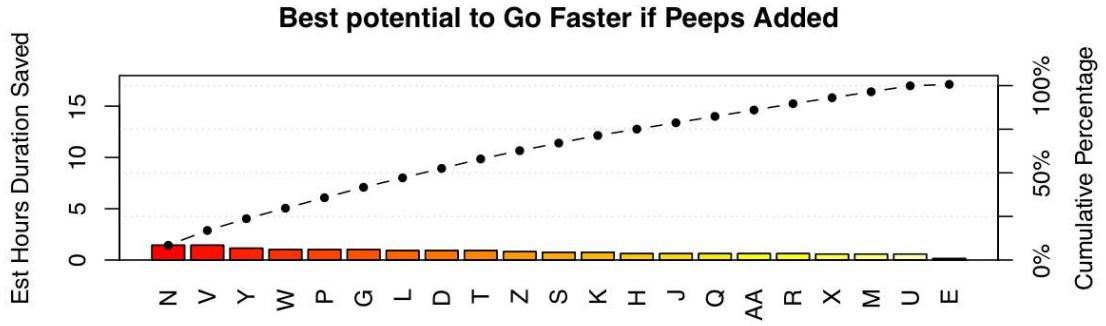
  This graph is a pareto of the POS values that is currently planned to be ”harvested” for each sector. The chart is given in decreasing order of sectors by POS and shows the contribution to total POS from each sector for only this portion of the search. This represents graphically the difference between the brown Start bar and the pink finish bar in the bar graph above. This chart effectively gives the order in which the search tasks (525’s) should be issued. The example given shows that sectors W, Z, and X have the greatest search potential and that in this round of planning approximately 45% POS is expected.

- Third Graph - POS Remaining by Sector for this Cycle of Planning

  This graph is a pareto of the POS values that are remaining after this plan is executed. These values represent the difference between the POC blue bar on the first graph and the ending pink bar. The pareto is sorted in order of decreasing POS remaining. In the example given the chart shows that if a subsequent search effort is planned, sectors E, W, and Z hold the greatest potential and that the most that approximately 35% of the POS remains unharvested.







Example Fourth, Fifth, and Sixth Graphs for Active Sectors

- Fourth Graph - Searchers Required by Sector for this Cycle of Planning

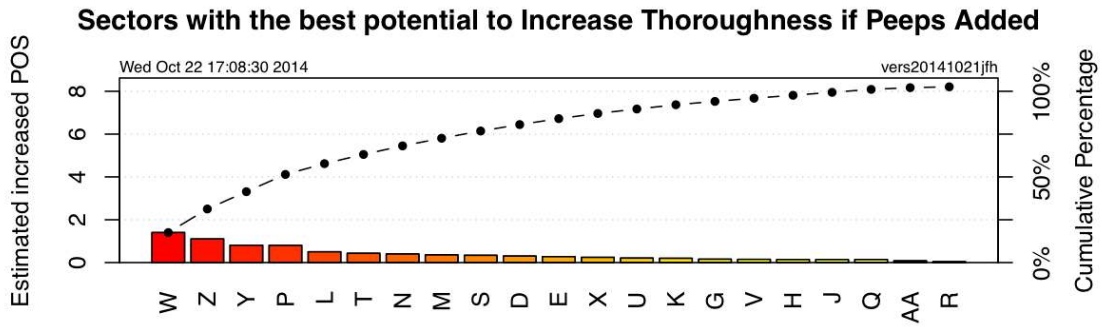
  This graph is a pareto of the number of searchers that the plan is based upon by sector, including a running total of searchers. The head count does not include supervisory support. In the example given, sectors X and H consume the greatest number of searchers and approximately 80 searchers will be required if all sectors are to be searched at the same time.

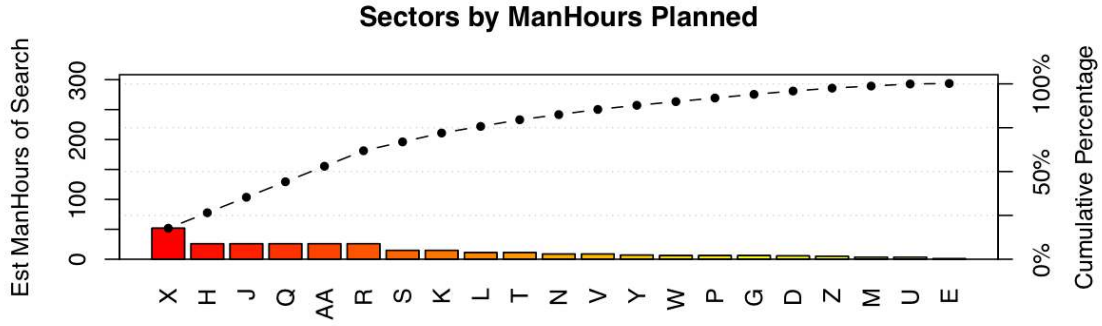
- Fifth Graph - Target Hours by Sector for this Cycle of Planning

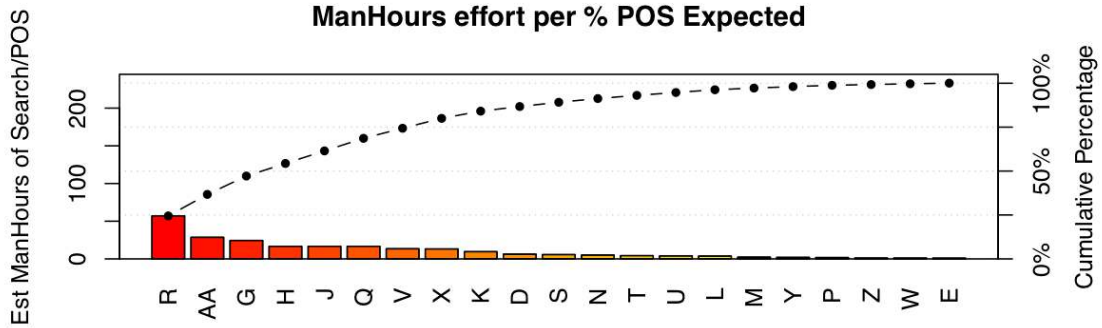
  This graph is a pareto of the duration of the search by sector, including a running total of duration hours. Duration hours are the hours it will take to clear the sector by the clock. In the example given, sectors X and N will take the longest time to search and if all sectors are searched sequentially it will take more than 70 hours to complete the effort.

- Sixth Graph - Best potential to Go Faster if Peeps Added

  This graph is a pareto of the time saved by sector if one additional person was added to each of the sectors listed. The chart is sorted by decreasing saved duration hours, so the first sector in the chart is the first team to consider adding a person to if the goal is to shorten the duration of the search. In this example, adding an additional searcher to sector N would result in the largest savings of time. So N is the first sector deserving of an added searcher if the goal is to shorten the duration of the search. The addition of one person per team would be expected to save over 15 duration hours across the plan.







Example Seventh, Eighth, and Ninth Graphs for Active Sectors

- Seventh Graph - Sectors with the best potential to Increase Thoroughness if Peeps Added

  This graph is a pareto of the POS saved by sector if one additional person was added to each of the sectors listed. The chart is sorted by decreasing saved duration hours, so the first sector in the chart is the first team to consider adding a person to if the goal is to increase the probability of success (POS). In the example, sector W would be the first team to consider adding a person to with a payout expected of over 1% POS. The addition of one person per team would add an additional 8% to the plan POS, raising it from 45% to approximately 54%.

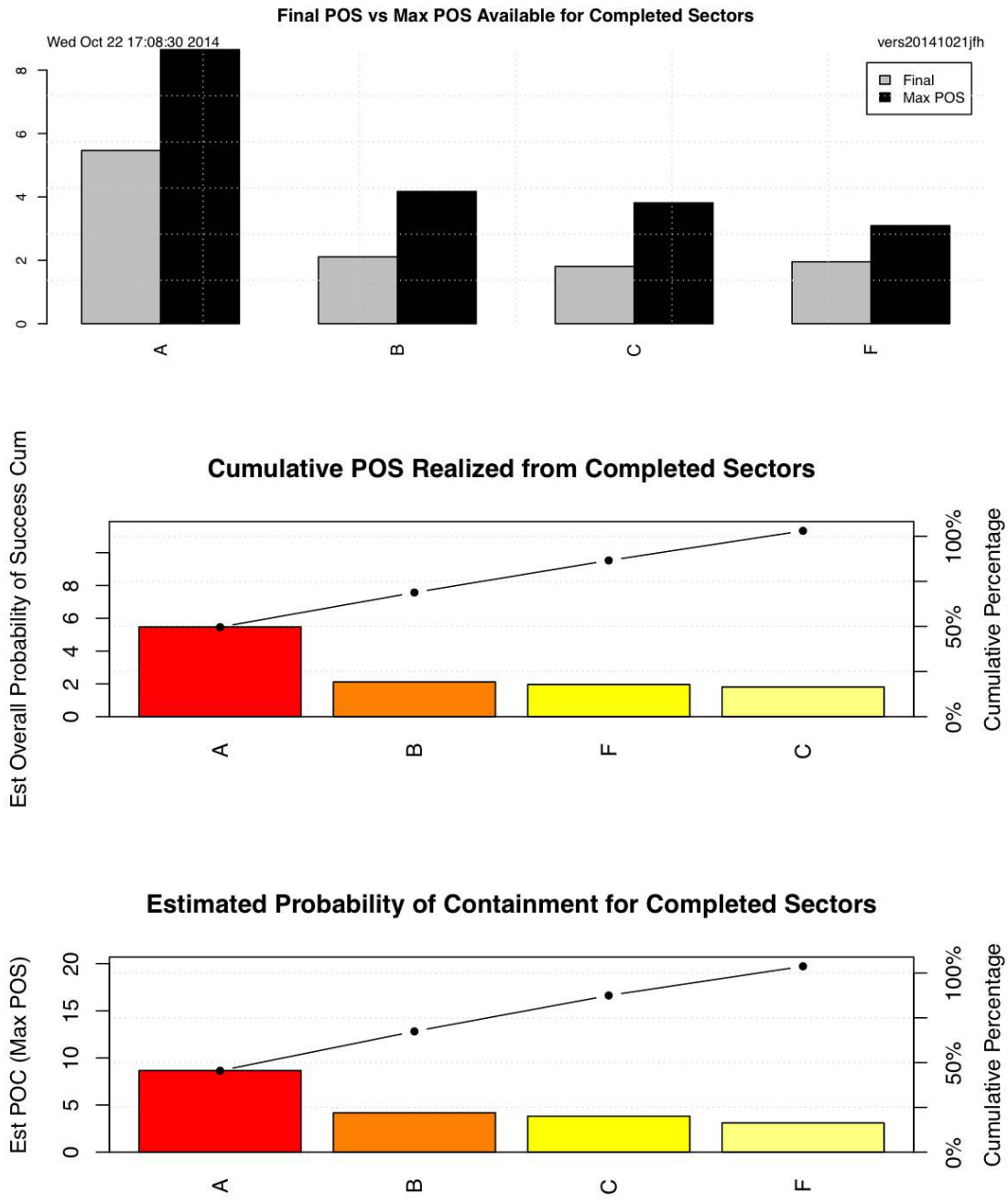
- Eighth Graph - Sectors by ManHours Planned

  This graph is a pareto of the total man hours required by sector to complete the search. The chart is sorted by decreasing man hours, so the first sector in the chart is the sector consuming the greatest effort in man hours. In the example given, sector X is the sector consuming the greatest number of man hours, with approximately 60 man hours planned to search that sector. Across all sectors over 800 man hours of searching is planned.

- Ninth Graph - ManHours effort per % POS Expected

  This graph is a pareto of the man hours of effort required divided by the POS returned from that effort. This metric identifies the sectors that require the greatest number of man hours to return the least amount of POS. In the example

shown, sectors R, AA, and G would be the first sectors to consider more rapid search techniques in. The total metric in this case is meaningless.



Example First, Second, and Third Graphs for Inactive Sectors

7.2 Inactive Sectors

For sectors which are no longer considered viable sectors for further search activity, the following graphs are produced. Inactive sectors have a negative POScum value (column L).

- First Graph - Final POS vs Max POS Available for Completed Sectors

  The first graph is a bar chart summary of the POC/POScum values for each sector that has been 'completed', sorted in decreasing order of POC. The final value is the value given for the POScum listed in the output file. This represents the amount of POS that has already been 'harvested' in earlier searches before the current plan is implemented. The max POS is the most POS that can be obtained from the sector and is the value of the POC. This bar chart summarizes the search results for the sectors where no further effort is planned. In this example, the chart shows that sectors A and B held the greatest potential, yielded approximately 66% of their value, and that no further work is planned for these sectors.

- Second Graph - Cumulative POS Realized from Completed Sectors

  This graph is a pareto of the POS values that have been ”harvested” from the completed sectors. The chart is given in decreasing order of sectors by POS and shows the contribution to total POS from each sector for only this portion of the search. This represents graphically the height of the gray bar in the bar graph above. This chart effectively gives the order of importance of the sectors searched in terms of the POS harvested and summarizes the POS obtained. The example given shows that sectors A, B, and F resulted in the greatest search potential and that approximately 12% of the POS has been consumed.

- Third Graph - Estimated Probability of Containment for Completed Sectors

  This graph is a pareto of the POC values for the sectors that have been completed. The pareto is sorted in order of decreasing POC. In the example given the chart shows that sectors A, B, and C hold the greatest potential and that the completed sectors account for approximately 20% of the POC.

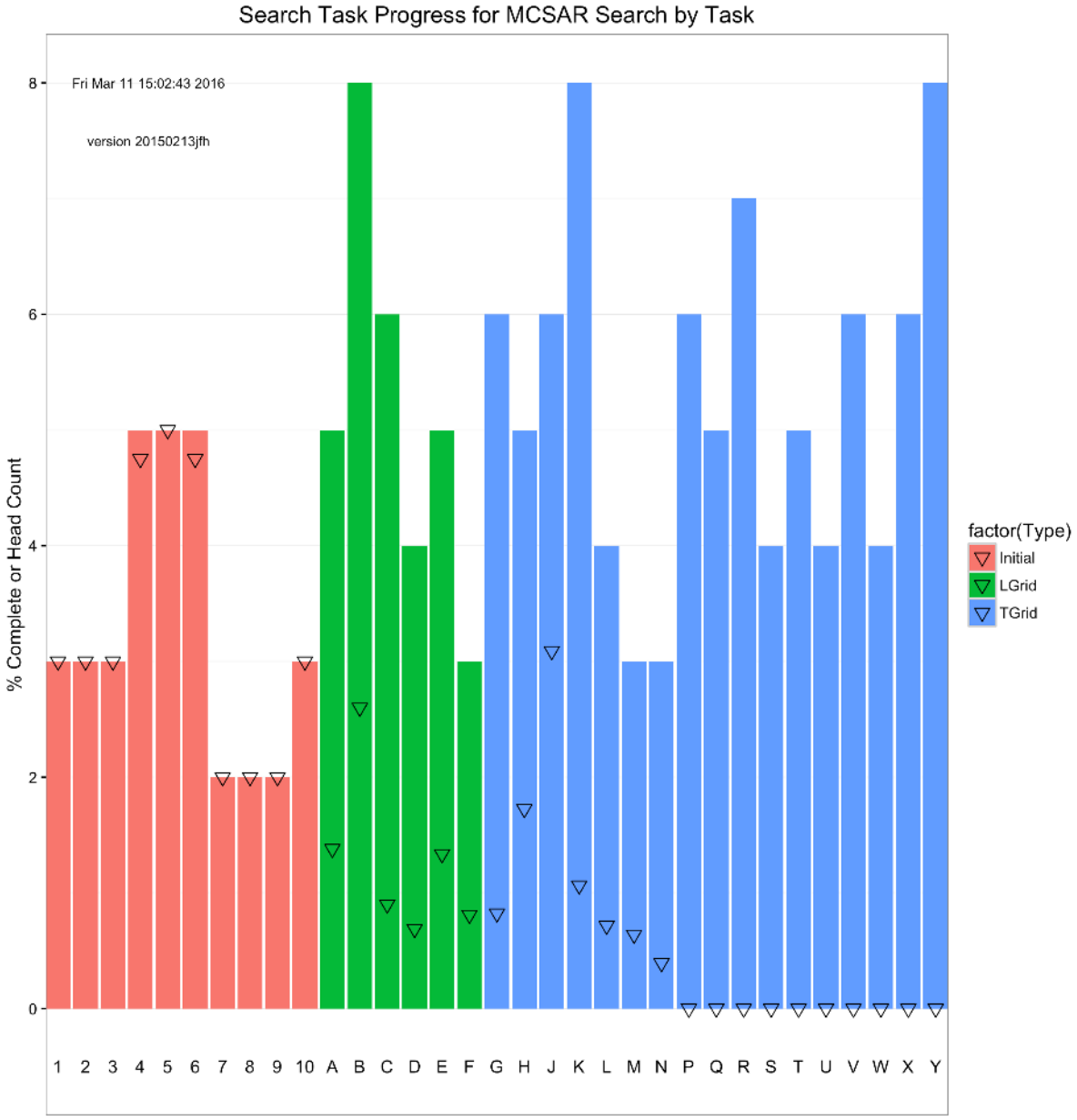
8 tracking() Use

8.1 Basic Operation

The tracking() function is a simple function to report against the progress of 525 Task Sheets issued to the field. Following the design of searchme(), a CSV file named ”Tracking.csv” must be stored in the working directory of R for this function to work properly. The ”Tracking.csv” spreadsheet has columns that track the progress of the search by 525 task type. The resulting graph is posted to search management and to staging to communicate the progress of the search within an operational period.

8.2 Graphical Output

The typical graphical output is displayed below. The graph shows progress towards completion of the task by way of the marking triangle, and the amount of resource tied up in the task by the height of the bar. As with other package graphs, each one is date/time stamped for the time of issue and each one displays the package version date.



Example of Tracking Graphical Output

8.3 Tracking.csv

Please see the example below for the columns of a typical "tracking.csv" file used for input with the tracking() function.

	A	B	C	D	E
1	Task	PerCentComplete	TSearchers	Type	Comment
2	IA	0	2	Initial	Residence
3	IB	100	2	Initial	Initial
4	IC	100	2	Initial	Residence
5	II	100	3	Initial	Initial
6	IM	100	8	Initial	Canvas
7	ID	66	3	Initial	AMDRs
8	IE	100	2	Initial	Track

Example of Tracking Input file in csv format

9 Many Thanks

Many thanks to Henrik Bengtsson for his assistance in helping me prepare this package for distribution. And Many thanks to the authors of the packages underpinning this package, including:

9.1 Bengtsson H (2015). R.rsp: Dynamic Generation of Scientific Reports. R package version 0.21.0, **https://github.com/HenrikBengtsson/R.rsp**.

9.2 Sarkar, Deepayan (2008) Lattice: Multivariate Data Visualization with R.

Springer, New York. ISBN 978-0-387-75968-5

9.3 Scrucca, L. (2004). qcc: an R package for quality control charting and

statistical process control. R News 4/1, 11-17.

9.4 R Core Team (2015). R: A language and environment for statistical

computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

9.5 Markus Helbig, Simon Urbanek and Ian Fellows (2013). JGR: JGR − Java GUI

for R. R package version 1.7-16. https://CRAN.R-project.org/package=JGR

9.6 H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer−Verlag

New York, 2009.