

A practical approach to subset selection with chance constraints

Thomas Monks ¹ Christine S.M Currie ²

¹NIHR CLAHRC Wessex, UoS / Alan Turing Institute

²CORMSIS, University of Southampton

Winter Simulation Conference 2018

Overview

- 1 Problem description
 - Motivation
 - Previous Work
- 2 Methods
 - Methods
- 3 Python Implementation
 - Setup
- 4 A practical problem
 - A practical problem

Motivation

Aim: support system experts making complex decisions involving **multiple objectives** and a **large number of scenarios** for the system

Factors:

- Some unquantifiable (e.g., political) variables
- Large, complex, slow-running simulation model
- Simulation practitioner without a PhD in statistics/simulation
- Off-the-shelf simulation package

Motivation

Aim: support system experts making complex decisions involving **multiple objectives** and a **large number of scenarios** for the system

Factors:

- Some unquantifiable (e.g., political) variables: Find a subset not a single optimum
- Large, complex, slow-running simulation model: Use variance reduction techniques, e.g., CRN
- Simulation practitioner without a PhD in statistics/simulation: Reduce the need for expert statistical judgment
- Off-the-shelf simulation package: Difficult to implement fully sequential methods

Motivation

Requirements

- R1 The choice of options to include in the experimentation and the number of replications to make can only be changed once during the experiment (two-stage method)
- R2 The procedure should not impose any distributional assumptions on the simulation output

Problem Description

Assume that we are comparing $k \geq 2$ systems and are primarily interested in minimizing the mean value of a particular output

$$x_i = \sum_{j=1}^n x_{ij} / n$$

where $i = 1, \dots, k$ but are also interested in L secondary outputs or objectives

$$y_{il} = \sum_{j=1}^n Y_{ijl} / n$$

Subset Selection with Chance Constraints

Aim: Identify a shortlist (subset) of systems (\mathbf{S}^*) that are all within a **proportion** β of the best system with probability $1 - \alpha$

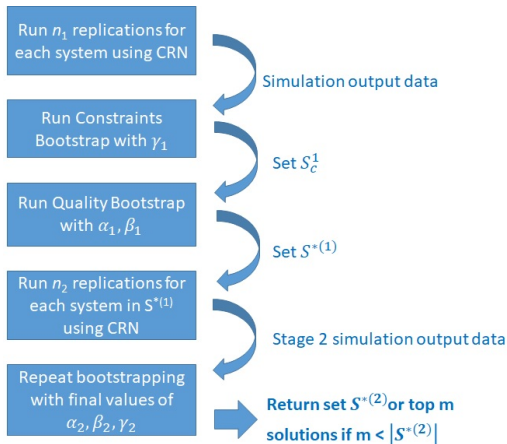
And satisfy the chance constraints with a probability $1 - \gamma$

We restrict the number of systems on the shortlist to $\min\{m, |S^*|\}$ by taking the **top m** systems that satisfy the above constraints

Previous Work

- 1 **Approach:** [Branke et al. 2007] suggest three categories: indifference zone, OCBA and Expected Value of Information
- 2 **Chance constraints** [Hong et al. 2015] suggest two approaches to dealing with chance constraints: Expectation Constrained Selection and Chance Constrained Selection.
- 3 **Subset selection** authors use either OCBA or indifference zone methods to maximize/guarantee the probability of correct selection of the best m of k systems

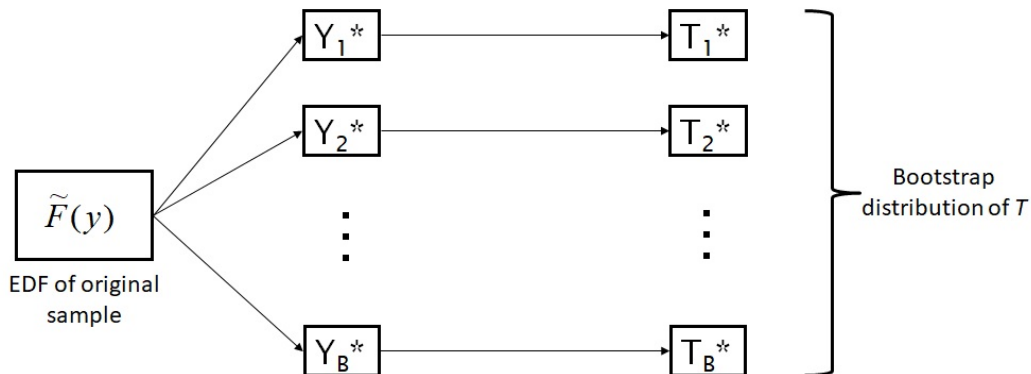
Big Picture



- Method relies on bootstrapping
- Set stage 1 parameters so that we are risk averse
- Balancing risk of missing a good solution versus including too many in stage 2
- Trade off between n_1 and n_2

Non-Parametric Bootstrapping

Resampling method used to infer properties for a set of data.



Constraints Bootstrap

Aim: Identify systems likely to violate the chance constraints

Constraints Bootstrap

- 1 Input a set of bootstrap samples $\mathbf{Y}^{*(1)}, \mathbf{Y}^{*(2)}, \dots, \mathbf{Y}^{*(B)}$ and for each calculate $y_l^{*(b)}, l = 1, \dots, L$.
- 2 Include systems in the final feasible set \mathbf{S}_c if

$$\frac{1}{B} \sum_{b=1}^B \prod_{l=1}^L I \left\{ y_l^{*(b)} \geq 0 \right\} \geq 1 - \gamma,$$

- 3 Return \mathbf{S}_c .

Quality Bootstrap

Aim: identify a set of systems with means within a distance β of the best system with probability $1 - \alpha$

Quality Bootstrap

- 1 Define a new variable $d_{ij} = x_j^* - x_{ij}$
- 2 Generate B bootstraps of the d_{ij}
- 3 In each bootstrap sample, identify systems with differences less than $\beta \bar{x}^*$, where \bar{x}^*

Quality Bootstrap

Quality Bootstrap (Cont'd)

- 4 Identify \mathbf{S}^* such that it is the biggest set for which

$$\frac{1}{B} \sum_{b=1}^B \prod_{j \in \mathbf{S}_c} I\{|d_{ij}^{*(b)}| \leq \beta \bar{x}^*\} \geq 1 - \alpha.$$

- 5 Return \mathbf{S}^*

Advice on installation of Python



The screenshot shows the Anaconda website's download page. At the top, there is a navigation bar with links for 'Anaconda Cloud', 'Documentation', 'Blog', and 'Contact', followed by a search icon. Below this, a secondary navigation bar includes 'What is Anaconda?', 'Products', 'Support', 'Community', 'About', and 'Resources', with a prominent green 'Download' button on the right. The main content area has a green background with a white geometric pattern. It features the 'ANACONDA' logo on the left and the heading 'Download Anaconda Distribution' in large white text. Below the heading, it states 'Version 5.1 | Release Date: February 15, 2018'. At the bottom, it says 'Download For:' followed by icons for Windows, macOS (Apple logo), and Linux (Tux penguin).

ANACONDA.

Anaconda Cloud Documentation Blog Contact

What is Anaconda? Products Support Community About Resources **Download**

Download Anaconda Distribution

Version 5.1 | Release Date: February 15, 2018

Download For:   

<https://www.anaconda.com/download/>

Code available from GitHub

CLAHRWessex / BootComp

Unwatch 2 Unstar 2 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Multiple comparison tool for simulation output using bootstrapping

Manage topics

53 commits 1 branch 0 releases 1 contributor MIT

Branch: master New pull request

Create new file Upload files Find file Clone or download

TomMonks Easy switch between single and parallel bootstrap

data	Debugged indexing and slicing of dataframe containing sy	
Images	Debugged indexing and slicing of dataframe containing sy	
results	Added rank_systems_min and rank_systems_max to Boot	
.gitignore	Initial commit	a year ago
BasicStatistics.py	Added proportion_x2_greaterthan_x1 function to Bootstrap. It calcul...	a year ago
BootChartExtensions.py	Added todo list	10 months ago
BootComp.py	Fixed bug with bs.msmlargest bs.mlargest zero indexing.	10 months ago

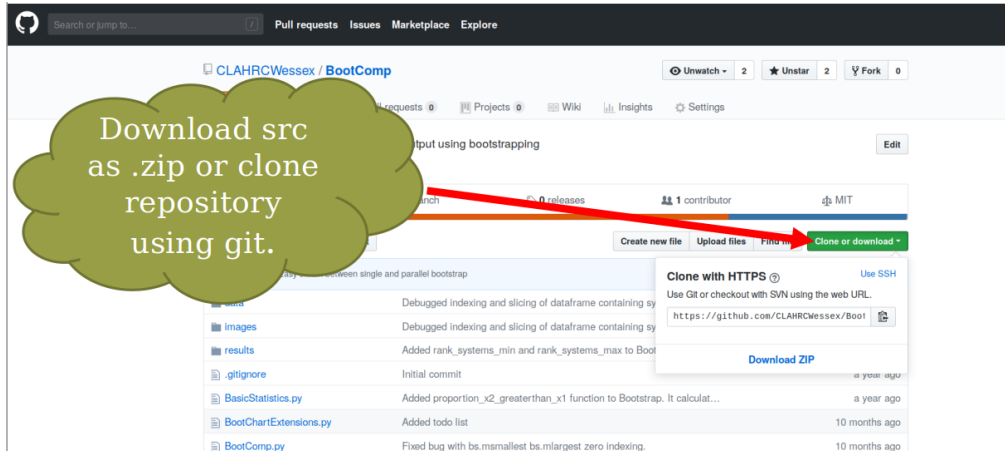
Clone with HTTPS Use SSH

Use Git or checkout with SVN using the web URL.

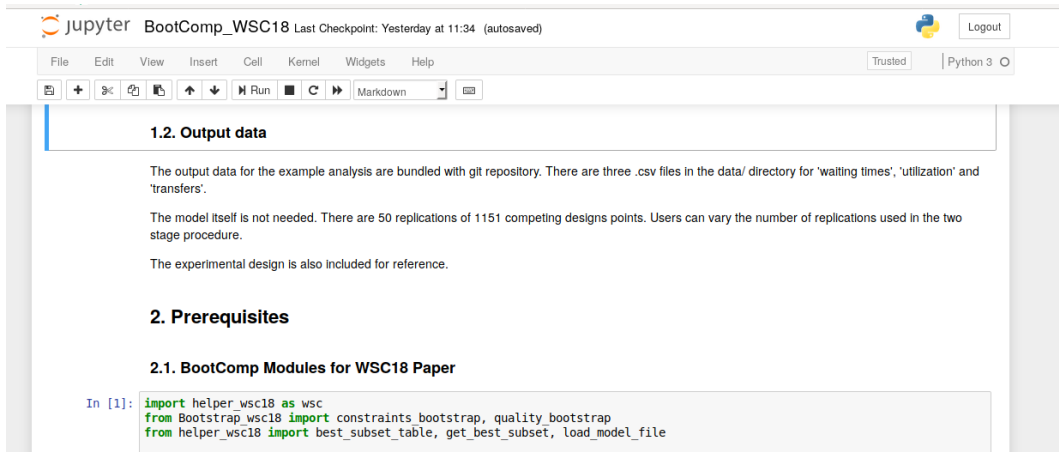
https://github.com/CLAHRWessex/Boot

Download ZIP

Code available from GitHub



Jupyter Notebook Implementation



The screenshot shows a Jupyter Notebook interface. The top bar includes the Jupyter logo, the notebook name 'BootComp_WSC18', and the last checkpoint information 'Last Checkpoint: Yesterday at 11:34 (autosaved)'. On the right, there is a 'Logout' button and a Python 3 environment selector. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar contains icons for file operations, running, and markdown. The main content area displays a markdown cell with the following text:

1.2. Output data

The output data for the example analysis are bundled with git repository. There are three .csv files in the data/ directory for 'waiting times', 'utilization' and 'transfers'.

The model itself is not needed. There are 50 replications of 1151 competing designs points. Users can vary the number of replications used in the two stage procedure.

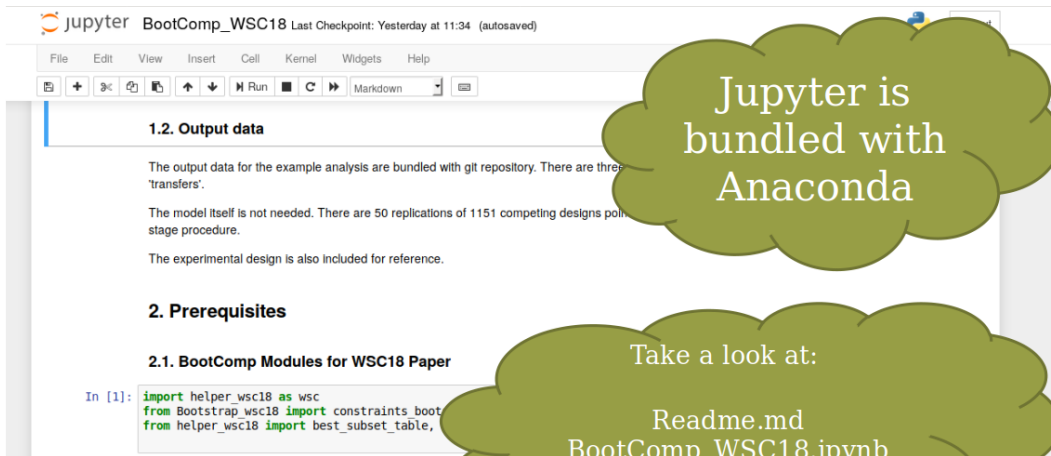
The experimental design is also included for reference.

2. Prerequisites

2.1. BootComp Modules for WSC18 Paper

```
In [1]: import helper_wsc18 as wsc
from Bootstrap_wsc18 import constraints bootstrap, quality bootstrap
from helper_wsc18 import best_subset_table, get_best_subset, load_model_file
```

Jupyter Notebook Implementation



The screenshot shows a Jupyter Notebook titled 'BootComp_WSC18' with a last checkpoint from yesterday at 11:34. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, navigation, and execution. The notebook content is in Markdown format, featuring two main sections: '1.2. Output data' and '2. Prerequisites'. The '1.2. Output data' section contains three paragraphs of text. The '2. Prerequisites' section includes a subsection '2.1. BootComp Modules for WSC18 Paper' which contains a code cell with Python import statements. Two green callout clouds are overlaid on the right side of the notebook. The top cloud contains the text 'Jupyter is bundled with Anaconda'. The bottom cloud contains the text 'Take a look at: Readme.md BootComp_WSC18.invnh'.

Jupyter BootComp_WSC18 Last Checkpoint: Yesterday at 11:34 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

1.2. Output data

The output data for the example analysis are bundled with git repository. There are three 'transfers'.

The model itself is not needed. There are 50 replications of 1151 competing designs point stage procedure.

The experimental design is also included for reference.

2. Prerequisites

2.1. BootComp Modules for WSC18 Paper

```
In [1]: import helper_wsc18 as wsc
from Bootstrap_wsc18 import constraints bootstrap
from helper_wsc18 import best_subset_table,
```

Jupyter is bundled with Anaconda

Take a look at:
Readme.md
BootComp_WSC18.invnh

Dependencies

- The readme.md provides an install guide (read the readme!)
- Create a conda environment
- Environments allow you to switch versions of Python packages to make sure you are using the same dependencies as the original code

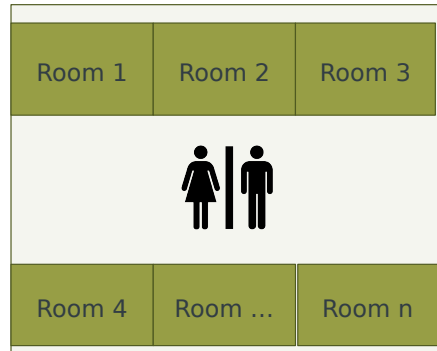
```
conda env create -f environment.yml  
conda activate bootcomp
```

Designing a rehabilitation ward



**Patients waiting in
an acute hospital for
transfer to
rehabilitation**

Transfer

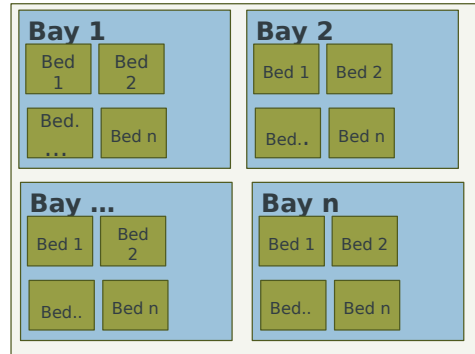


A practical problem

Designing a rehabilitation ward



Transfer



A practical problem

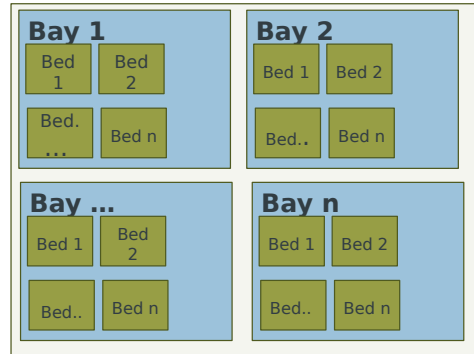
Designing a rehabilitation ward



Transfer

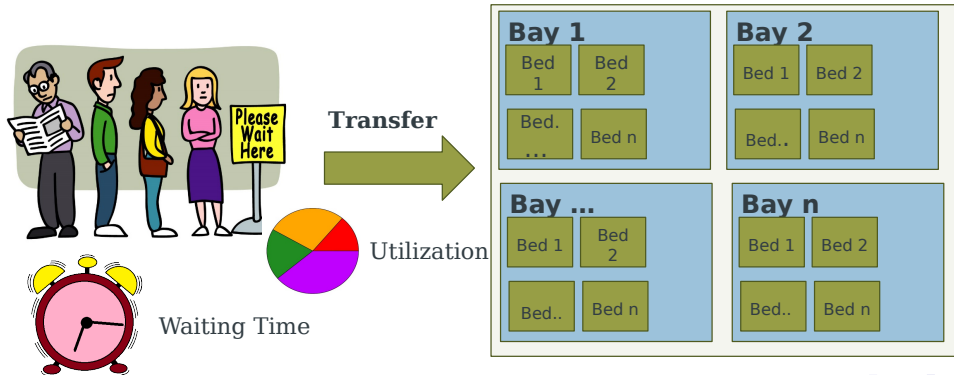


Waiting Time



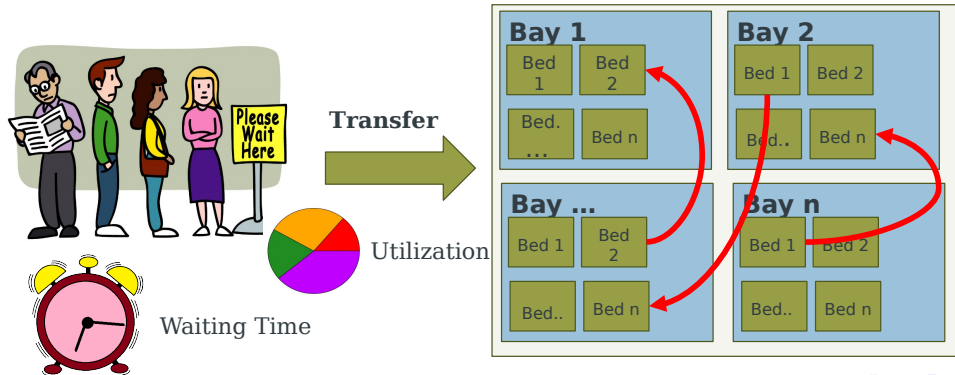
A practical problem

Designing a rehabilitation ward



A practical problem

Designing a rehabilitation ward



Design of Experiments

1051 competing designs

Decision Variables

- No. beds
- Bay size + no. bays
- No. single rooms

Chance Constraints

- Utilization of beds
- Patient transfers between bays

Further work

- Comparison with sequential budget allocation algorithms finding top-m systems with a **single** performance measure
 - Set of 10 normal distributions $N(i, 6)$
 - Set of 100 normal distributions $N(i/100, 6)$
 - Law inventory example
- Comparisons using common random numbers
- Identifying “good” values for the parameters in the first stage: $N_0, \alpha_1, \beta_1, \gamma_1$

References



Jürgen Branke, Stephen E. Chick and Christian Schmidt (2007)

Selecting a selection procedure

Management Science 53, 1916–1932



L. Jeff Hong, Jun Luo and Barry L. Nelson (2015)

Chance constrained selection of the best.

INFORMS Journal on Computing 27, 317 – 334.