C++ Example

```
// Scanner class definition
class Scanner {

// Declare that this class is friends with Movement.
friend class Movement;

private:

   // The channel numbers
   unsigned int end_channel, start_channel;

   // Positional coordinates
   float pan, tilt;

   // Color and gobo settings
   unsigned int color, gobo;

   // Channel mapping values
   unsigned int color_channel, gobo_channel, pan_channel, tilt_channel;


public:

   // The constructor function, which takes
   // a pointer to a scene frame object -> MOVE TO CORE
   Scanner(Scene *, Geometry *);

   // The pointer for which scene the
   // scanner is associated with -> MOVE TO CORE
   Scene * scene_associate;

   // The pointer for which geometry the
   // scanner is associated with. -> MOVE TO CORE
   Geometry * geometry_associate;

   // Channel setting functions -> MOVE TO CORE
   void set_start_channel(unsigned int);
   void set_end_channel(unsigned int);

   // Positional functions
   void set_pan(unsigned int, unsigned int);
   void set_pitch(unsigned int, unsigned int);

   // Color and gobo functions
   void set_color(unsigned int, unsigned int);
   void set_gobo(unsigned int, unsigned int);

   // The function to map channels -> MOVE TO CORE
```

```
    void map_channels(vector<int>, vector<string>);

    // Write to frames function -> MOVE TO CORE/Shared
    void write_frames(unsigned int, unsigned int, unsigned int);

};

/* Constructor definition */

Scanner::Scanner(Scene * scene_ptr, Geometry * geo_ptr) {

    //cout << &scene_ptr;
    // Set the scene association.
    //this.scene_associate->scene_ptr;
    this->scene_associate = scene_ptr;
    this->geometry_associate = geo_ptr;

}

void Scanner::write_frames(unsigned int frame_number, unsigned int channel_number, unsigned int
value) {

    //cout << "frame_number: " << frame_number << "\n";
    //cout << "channel_number: " << channel_number << "\n";
    //cout << "value: " << value << "\n";

    // Can adjust output based on which channel we want to view.
    if(channel_number == 7) {
        cout << "INFO: " << frame_number << " " << channel_number << " " << value << "\n";
    }

    // Write to the frame using the relevant channels.
    this->scene_associate->frames[frame_number][channel_number] = value;

    // Print the frame out.
    //v_print(this->scene_associate->frames);

}

/* Scanner function definitions. */

// Map the channels.

// Note:  Couldn't use a multidimensional vector here
// since data types have to match between dimensions.
void Scanner::map_channels(vector<int> channels, vector<string> channel_function) {

    // Since the relationship between channels and their
```

```
   // functions is one-to-one, we can just loop over
   // the channels and functions simultaneously, using
   // the base mappings to convert strings to object names.

   //cout << this->pan_channel << " " << this->pitch_channel << " " << this->color_channel << " " << this-
>gobo_channel << "\n\n";

   vector<int>::const_iterator row;

   for (vector<double>::size_type i = 0; i < channels.size(); i++)
   {
      // The channel.
      //cout << *row << "\n";
      //cout << channels[i] << " " << channel_function[i] << "\n";

      // Set the values.

      if(channel_function[i] == "color_channel") {
         this->color_channel = channels[i];
      } else if(channel_function[i] == "gobo_channel") {
         this->gobo_channel = channels[i];
      } else if(channel_function[i] == "pan_channel") {
         this->pan_channel = channels[i];
      } else if(channel_function[i] == "tilt_channel") {
         this->tilt_channel = channels[i];
      }
   }

   cout << this->pan_channel << " " << this->tilt_channel << " " << this->color_channel << " " << this-
>gobo_channel << "\n\n";

}

// Set the channels.
void Scanner::set_start_channel(unsigned int start_channel_val) {

   this->start_channel = start_channel_val;

}

void Scanner::set_end_channel(unsigned int end_channel_val) {

   this->end_channel = end_channel_val;

}

// Set the positions.
void Scanner::set_pan(unsigned int frame_number, unsigned int pan_val) {
```

```
    // Set the pan using the relevant channel.
    // We could write directly to the frame with pan_val,
    // but it is desirable to set the state of the light
    // for debugging purposes.  Same with the following
    // functions.
    this->pan = pan_val;

    // Write to the frame.
    write_frames(frame_number, this->pan_channel, this->pan);

}

void Scanner::set_pitch(unsigned int frame_number, unsigned int pitch_val) {

    // Set the tilt using the relevant channel.
    this->tilt = pitch_val;

    // Write to the frame.
    write_frames(frame_number, this->tilt_channel, this->tilt);

}

// Set the color and gobo information.
void Scanner::set_color(unsigned int frame_number, unsigned int color_val) {

    // Set the color using the relevant channel.
    this->color = color_val;

    // Write to the frame.
    write_frames(frame_number, this->color_channel, this->color);

}

void Scanner::set_gobo(unsigned int frame_number, unsigned int gobo_val) {

    // Set the gobo using the relevant channel.
    this->gobo = gobo_val;

    // Write to the frame.
    write_frames(frame_number, this->gobo_channel, this->gobo);

}

#include <iostream>

// For vectors and printing
#include <vector>
```

```
#include <algorithm>

// For string concatenation
#include <numeric>

// System includes
#include <windows.h>

using namespace std;

// Core (engine) class definition
class Core {
private:

   /* ----- Preliminaries ----- */

   // The screen clearing function
   void screen_clear() {
      system("cls");
   }

   // The vector printing function
   void v_print(vector<string> &incoming) {
      for (vector<string>::const_iterator i = incoming.begin(); i != incoming.end(); ++i) {
         cout << *i;
      }
   }

   // The file listing function
   void list_files(vector<string> directory, vector<string> file_type) {

      cout << "File listing (.";
      v_print(file_type);
      cout << ") for videos\n\n";

      // Concatenate the vectors so that we can find the files.
      vector<string> concat = directory;

      concat.push_back("*.");
      concat.insert(concat.end(), file_type.begin(), file_type.end());

      string concat_check;
      for (vector<string>::const_iterator i = concat.begin(); i != concat.end(); ++i)
         concat_check += *i;

      // List all the files in the video directory matching file_type.
      WIN32_FIND_DATA data;
      //HANDLE hFind = FindFirstFile("C:\\..\\videos\\*.mp4", &data);
```

```
    HANDLE hFind = FindFirstFile(concat_check.c_str(), &data);

    if ( hFind != INVALID_HANDLE_VALUE ) {
       do {
          cout << data.cFileName << endl;
       } while (FindNextFile(hFind, &data));
       FindClose(hFind);
    }

  }

  /* ----- Show Manager ----- */

  // The main menu function
  unsigned int main_menu(vector<string> &mm_current_show) {

    /* -------- MAIN MENU -------- */

    // Clear anything out that might be lingering.
    screen_clear();

    // The header.
    header();

    cout << "MAIN MENU\n\n";
    cout << "1)  Load show file (.pm)\n";
    cout << "2)  Create new show\n\n";

    cout << "Current show: ";
    v_print(mm_current_show);
    cout << "\n\n";

    cout << "Please select an option: ";

    // Use only integers for options.
    unsigned int option;
    cin >> option;
    cin.ignore();

    // See what we've got.
    switch(option) {

       case 1: /*system("cls"); menu(*load_pm());*/ break;
       case 2: create_show(); break;
       default: /*cout << "Invalid option selected, returning to menu!"; Sleep(2000); system("cls");
main_menu(vector<string> temp (4) = "None");*/ break;

    }
```

```
    return 0;

}

// The show manager function
void show_manager() {

    // The show manager will first check the
    // pm.conf file in the directory.  Next,
    // if no default show is specified in the
    // conf, then it will load all shows (.pm files)
    // from the working directory.

    // TO BE INSERTED LATER

    // If this fails, then create a "blank" show manager.
    vector<char> shows;

    // The current show.
    vector<string> current_show;

    // Set the current show to none.
    //current_show.resize(3);

    current_show.push_back("None");

    // Now enter the main logic.  Create a variable
    // to keep the program open.

    bool exit_check = false;

    while(exit_check == false) {

        // Call the menu function with the current show.
        main_menu(current_show);

    }

    //char name[50] = "the scene";

    //create_scene(3, name, 2000);

}

struct Scene {
    unsigned int scene_id;
    char * scene_name;
```

```
        double duration;
        unsigned int attached_fixture_ids[512];
    };

public:

    // The class constructor function
    Core();

    // Show creation function
    void create_show();

    // Scene creation function
    void create_scene(unsigned int, unsigned int, char *, double);

    // Fixture assignment function
    void assign_output(unsigned int scene_identifier, char fixture_identifier);

};


/* Core function definitions. */

// Create a show manager when the class constructor is called.
Core::Core(void) {

    show_manager();

}


// Create a show.
void Core::create_show() {

    // Clear the screen.
    screen_clear();

    // The header.
    header();

    // Display information to the user and get input.
    cout << "CREATE NEW SHOW\n----------\n\n";

    vector<string> looking_directory;
    looking_directory.push_back("C:\\...\\videos\\");

    vector<string> file_extension;
    file_extension.push_back("mp4");
```

```
    list_files(looking_directory, file_extension);

    //list_options

}

// Create a scene.
void Core::create_scene(unsigned int show_id, unsigned int scene_id, char scene_name[], double
duration) {

    // Here we instantiate a new scene.
    Scene new_scene;

    // Set the relevant variables in the scene.
    new_scene.scene_id = scene_id;
    new_scene.scene_name = scene_name;
    new_scene.duration = duration;

    // Now call the assign output

    cout << "scene_id: " << new_scene.scene_id << endl;
    cout << "scene_name: " << new_scene.scene_name << endl;

}

// Assign fixtures to a scene.
void Core::assign_output(unsigned int scene_identifier, char fixture_identifier) {

    // Here we attach individual lights to scenes.

}



int main()
{
    // Create a core.
    Core new_core;

    // Test variables.
    unsigned int scene_number = 24;
    char scene_naming[100] = "test name";
    double duration_out = 3000;

    new_core.create_scene(scene_number, scene_naming, duration_out);
    cout << "Hello world!" << endl;
    return 0;}
```

JavaScript Example

```
function update_lists(selected_list, un_pick) {

            // This function updates the select lists
            // based on items that have been clicked
            // in any of the other lists.

            // Could have passed a parameter indicating
            // which thing to update but that was too much work.

            // Determine which things do not equal "All".
            // Could probably clean this up so that a
            // similar logic is used as that being applied
            // to the search buttons.

            // Note that this loop logic is probably not
            // necessary since we're using a "picked"
            // value that can only be applied to one
            // select box at a time.  Essentially,
            // sub-setting is occurring with each
            // subsequent search selection.

            // The loop logic here is fast, consider
            // implementing elsewhere.

            $("#results").hide();

            var search_values = [], bulk_search_values =
$("#search_layout .search_option"), display_values = [];


            // Set only the selected box to "picked" and all
            // others to "not_picked".

            var a = 0, run = bulk_search_values.length;

            for (a; a < run; a++) {
                if ($(bulk_search_values).slice(a).attr("name") ==
selected_list && un_pick != true) {
                    $(bulk_search_values).slice(a).removeClass();
                    $(bulk_search_values).slice(a).addClass("picked
search_option");
                } else {
                    $(bulk_search_values).slice(a).removeClass();
                    $(bulk_search_values).slice(a).addClass("not_picked
search_option");
                }
            }


            // Get only the search values for picked.

            var picked_search =
document.getElementsByClassName("picked");
```

```
            if (picked_search.length > 0) {
                var i = 0, go = picked_search[0].length;

                for (i; i < go; i++) {
                    if (picked_search[0].options[i].selected == true &&
picked_search[0].options[i].value == "All") {
                        break;
                    } else if (picked_search[0].options[i].selected ==
true && picked_search[0].options[i].value != "All") {
                        search_values.push("\"" +
picked_search[0].options[i].value + "\"");
                    }
                }
            }


            // Get what's being displayed.

            $("#display_buttons td input").each(function () {
                if ($(this).is(':checked')) {
                    display_values.push($(this).val());
                }
            });

            display_values = display_values.join();


            // Now that we know where things have been selected,
            // go through and find the name and values so that
            // we can query SQL.


            if (search_values != "") {

                constructed = selected_list + "=" + search_values +
"&display=" + display_values;

                //alert(constructed);

                // Now call AJAX.  Here only the helper
                // table in SQL needs to be updated, then
                // initialize() is called.

                       $.ajax({url: "side_table.php?" +
constructed,}).done(function(res) {
                            document.getElementById("results").innerHTML
= res;
                    initialize(false);
                       });
            } else {
                reset(false);
            }

                 $("#results").show();
```

```
        }

        function side_revenue() {

            // This is for updating field values when
            // revenue or net assets are searched after
            // the city.

            var revenue_bottom =
document.getElementById("revenue_bottom").value,
                revenue_top =
document.getElementById("revenue_top").value,
                net_assets_bottom =
document.getElementById("net_assets_bottom").value,
            net_assets_top =
document.getElementById("net_assets_top").value, financial = 0, sending;


            // Set the range indicators

            if (revenue_bottom != "") {
                revenue_bottom = revenue_bottom + "," +
$("#revenue_bottom_bracket").val();
                financial = true;
            }

            if (revenue_top != "") {
                revenue_top = revenue_top + "," +
$("#revenue_top_bracket").val();
                financial = true;
            }

            if (net_assets_bottom != "") {
                net_assets_bottom = net_assets_bottom + "," +
$("#net_assets_bottom_bracket").val();
                financial = true;
            }

            if (net_assets_top != "") {
                net_assets_top = net_assets_top + "," +
$("#net_assets_top_bracket").val();
                financial = true;
            }


            // Now that we know where things have been selected,
            // go through and find the name and values so that
            // we can query SQL.

            if (financial != "0") {
                sending = "side_rna.php?rb=" + revenue_bottom + "&rt=" +
revenue_top + "&nab=" + net_assets_bottom + "&nat=" + net_assets_top;
            } else {
                sending = "reset_rna.php";
```

```
            }

                 $.ajax({url: sending,}).done(function(res) {
                document.getElementById("results").innerHTML = res;
                       //alert("here it is");
                       initialize(false);
                 });
        }

        function showUser() {
            // This is a cleaner solution then generating
            // unnecessary HTML on the page.  NOTE THAT
            // THIS CAN BE SIMPLIFIED TO SIMPLY SELECT
            // THE ACTIVE RECORDS FROM THE TABLE USING
            // A SIMPLE SORT.

            // Note that this is being called twice now when
            // a search button other than all is checked
            // and then unchecked.

            // Set sort parameters.

            // Get the info from sort_div.  The default sort
            // is by EIN ascending.

            var sort_by = "revenue", sort_direction = "DESC", sort_bulk =
$("#sort_div").attr("data-value");
            //alert(sort_bulk);
            if (sort_bulk != "") {
                // See if it's a simple sort.

                if (sort_bulk.indexOf("SIMPLE") > -1) {
                    // Split on the hyphen.

                    sort_bulk = sort_bulk.split("-")[1].split(",");
                    sort_by = sort_bulk[0];
                    sort_direction = sort_bulk[1];
                } else {
                    // Keep only the columns and
                    // sort orders.

                    var converted = $("#sort_div").attr("data-
value").substring(0, $("#sort_div").attr("data-value").length -
1).split(","), sort_length = converted.length / 3;
                    var temp_array_sb = [], temp_array_sd = [], i = 1;

                    for (i; i <= sort_length; i++) {
                        temp_array_sb.push(converted[3 * i - 3]);
                        temp_array_sd.push(converted[3 * i - 2]);
                    }

                    sort_by = temp_array_sb.join(",");
                    sort_direction = temp_array_sd.join(",");
                }
```

```
            }

            var lookup = document.getElementById("lookup").value;


            // Get which columns we are going to search in.

            var search_options;


            // Loop over all of the radio buttons.

            var i = 1, loop_length = $("#search_layout td").length;

            for (i; i < loop_length; i++) {
                if (document.forms["search"].elements[i].type ==
"checkbox") {
                    if (document.forms["search"].elements[i].checked ===
true) {
                        if (search_options === undefined) {
                            search_options =
document.forms["search"].elements[i].value;
                        } else {
                            search_options = search_options + '-' +
document.forms["search"].elements[i].value;
                        }
                    }
                }
            }


            // Get which columns we are displaying.

            var display_options;


            // Loop over all of the radio buttons.

            i = 0, loop_length = $("#display_buttons td").length;

            for (i; i < loop_length; i++) {
                if (document.forms["display"].elements[i].checked ===
true) {
                    if (display_options === undefined) {
                        display_options =
document.forms["display"].elements[i].value;
                    } else {
                        display_options = display_options + '-' +
document.forms["display"].elements[i].value;
                    }
                }
            }
```

```
            $.ajax({url: "query_table.php?q=" + lookup + "&display=" +
display_options + "&search=" + search_options + "&sorting=" + sort_by +
"&sort_order=" + sort_direction,}).done(function(res) {
                document.getElementById("results").innerHTML = res;

                        // If ein isn't selected, then hide the ein
column.

                        if (!($("#display_buttons
input").eq(1).prop("checked"))) {
                                // Hide ein

                                $('#selectable tr > td:nth-child(1),
#selectable tr > th:nth-child(1)').hide();
                        }
                        //alert("running");

                        //alert("stop");
                        // Clear the sort.td:not(:first)

                        maintain_rows();
                 });
        }
```

Python Example

```
from os import remove, chdir, mkdir
from os.path import exists
from shutil import copytree, rmtree
from PIL import Image
from subprocess import call
from pytesser import *
from PyPDF2 import PdfFileReader, PdfFileWriter

from sys import argv, exit
from getopt import getopt, GetoptError

from pymysql import connect

# from basic import wait

from urllib import URLopener, urlopen, urlretrieve
from urllib2 import urlopen, HTTPError, URLError

# from pprint import pprint
# from datetime import datetime

# import requests
from requests import get

from warnings import filterwarnings


def get_pdf_link(ein_send, fye):
    # Here we'll try to find the PDF we need to download.
    # Start by constructing the default URL.
    default = 'http://990s.foundationcenter.org/990_pdf_archive/' +
ein_send[0:3] + '/' + ein_send + '/' \
              + ein_send + '_' + fye + '_990.pdf'

    print 'Trying url: ' + default

    # We'll have a variable to keep track of the URLs.
    # We'll assume that it exists by default.
    url_found = 'true'

    try:
        urlopen(default)
    except HTTPError:
        url_found = 'false'
    except URLError:
        url_found = 'false'
    finally:
        if url_found == 'true':
            # The link is valid, so create a directory in which to store the
PDF.
            # Change the working directory.
            chdir('../../')

            # Delete the directory for the ein if it exists.
            if exists(ein_send):
                rmtree(ein_send)
```

```
            # Create the directory for this ein.
            mkdir(ein_send)

            # Download it.
            urlretrieve(default, ein_send + '\\' + ein_send + '.pdf')

            # See how many pages the PDF has.
            pdf_object = PdfFileReader(open(ein_send + '\\' + ein_send +
".pdf", 'rb'))
            number_of_pages = pdf_object.getNumPages()

            retrieve(ein_send, number_of_pages, pdf_object)
        else:
            # Write to the log file.
            print default + ' not found.'

def convert(file_name, ein_stuff_2):
    #print file_name
    # Convert the pdf to an image then read the text.
    #call('convert -density 300 ' + file_name + ' ' +
file_name.replace('.pdf', '.png'), shell=True)
    print 'C:\PROGRA~1\IMAGEM~1.3-Q\convert -density 300 ' + file_name + ' '
\
          + file_name.replace('.pdf', '.png')
    call('C:\PROGRA~1\IMAGEM~1.3-Q\convert -density 300 ' + file_name + ' '
         + file_name.replace('.pdf', '.png'), shell=True)
    print 'file_name.replace: ' + file_name.replace('.pdf', '.png')
    print 'folder_helper: ' + ein_stuff_2
    return image_to_string(Image.open(file_name.replace('.pdf', '.png')),
folder_helper=ein_stuff_2).lower()

def basic_process(page_number, x1, x2, seed, height, checking_for,
page_bulk, mode, ein_stuff):

    # Get the page at page_number and crop it.
    page_object = page_bulk.getPage(page_number)
    upper_left_y = int(page_object.trimBox.getUpperLeft_y())

    # Set y1 and y2.
    y1 = upper_left_y - seed
    y2 = y1 + height
    page_object.mediaBox.upperRight = (x2, y2)
    page_object.mediaBox.lowerLeft = (x1, y1)

    output = PdfFileWriter()
    output.addPage(page_object)
    # file_sending = ein_stuff + "\\test_export_" + str(page_number) +
".pdf"
    file_sending = "c:\\users\\ca\\drop\\" + ein_stuff + "\\test_export_" +
str(page_number) + ".pdf"
    output.write(file(file_sending, "wb"))

    if mode == "loose":
        if any(x in convert(file_sending, ein_stuff) for x in checking_for):
            # Delete the files since we're done with them.
            remove(file_sending)
            remove(file_sending.replace('.pdf', '.png'))
```

```
            return "true"
        else:
            # Delete the files since we're done with them.
            remove(file_sending)
            remove(file_sending.replace('.pdf', '.png'))
            return "false"
    elif mode == "tighter":
        if all(x in convert(file_sending, ein_stuff) for x in checking_for):
            # Delete the files since we're done with them.
            remove(file_sending)
            remove(file_sending.replace('.pdf','.png'))
            return "true"
        else:
            # Delete the files since we're done with them.
            remove(file_sending)
            remove(file_sending.replace('.pdf', '.png'))
            return "false"
    elif mode == "strict":
        if convert(file_sending, ein_stuff).strip() == str(checking_for[0]):
            # Delete the files since we're done with them.
            remove(file_sending)
            remove(file_sending.replace('.pdf', '.png'))
            return "true"
        else:
            # Delete the files since we're done with them.
            remove(file_sending)
            remove(file_sending.replace('.pdf', '.png'))
            return "false"

def retrieve(ein_info, page_count, bulk):

    # Set up a list of pages to print.
    printing = []

    # Try page 8 first.  See if it says "See Additional Data Table" or
    # if it's blank.

    if (basic_process(7, 0, 150, 422, 18, ['see', 'additional', 'data',
'table'], bulk, "loose", ein_info) == "true") \
            or (basic_process(7, 0, 150, 422, 18, [''], bulk, "strict",
ein_info) == "true"):

        #print "test 1"
        #print basic_process(7, 0, 150, 422, 18, ['see', 'additional',
'data', 'table'], bulk, "loose", ein_info)
        #print "test 2"
        #print basic_process(7, 0, 150, 422, 18, [''], bulk, "strict",
ein_info)

        # Didn't find the page so keep going.
        page_found = "false"
        page_counter = 8
        #print "not printing 7 and 8"
        while page_found == "false":
            #print "page counter " + str(page_counter)
            if basic_process(page_counter, 0, 200, 30, 25,
['additional','data'], bulk, "loose", ein_info) == "true":
```

```
                # Found the page.
                page_found = "true"

                # Add the pages to printing.
                printing.append(page_counter)
                printing.append(page_counter + 1)

            else:

                # Go to the next page.
                if page_counter < page_count-1:
                    page_counter += 1
                else:
                    page_found = "true"
    else:

        # Add this page and the next to append.
        printing.append(7)
        printing.append(8)
        #print "printing 7 and 8"
        page_counter = 9

    #page_counter += 1

    # Now look for the executive pay. Start by looking for "Schedule J".
    page_found = "false"

    page_counter = 21

    while page_found == "false":
        #print "Reading page " + str(page_counter) + " of " + str
(page_count - 1)
        if ((basic_process(page_counter, 0, 150, 30, 25, ['additional',
'data'], bulk, "loose", ein_info) == "true") and
                (basic_process(page_counter, 0, 150, 195, 43, ['(a) name'],
bulk, "strict", ein_info) == "true")) or \
                ((basic_process(page_counter, 0, 150, 26, 25, ['schedule'],
bulk, "loose", ein_info) == "true") and
                     (basic_process(page_counter, 0, 150, 100, 17, ['name',
'title'], bulk, "loose", ein_info) == "true")):

            page_found = "true"

            # Add the pages to printing.
            printing.append(page_counter)
            printing.append(page_counter + 1)

        else:
            # Go to the next page.
            if page_counter < page_count-1:
                page_counter += 1
            else:
                page_found = "true"

    # Print the pages.
    #print str(ein_info) + ".pdf"
```

```
    incoming = PdfFileReader(open(ein_info + '\\' + ein_info+".pdf", 'rb'))
    outgoing = PdfFileWriter()
    for page in printing:
        outgoing.addPage(incoming.getPage(page))
    #stream = file("c:\\users\\phaedrus\\desktop\\pet
projects\\prm\\pdf_out\\"+ein_info+".pdf", "wb")

    # PRM edit
    # Copy and paste materials to the shared folder...
    if exists("c:\\data\\wizsyn~1\\Dump\\"+ein_info+".pdf"):
        remove("c:\\data\\wizsyn~1\\Dump\\"+ein_info+".pdf")

    if exists("c:\\data\\wizsyn~1\\Dump\\"+ein_info):
        rmtree("c:\\data\\wizsyn~1\\Dump\\"+ein_info)

    stream = file("c:\\data\\wizsyn~1\\Dump\\"+ein_info+".pdf", "wb")
    copytree("c:\\users\\ca\\drop\\"+ein_info,
"c:\\data\\wizsyn~1\\Dump\\"+ein_info)
    outgoing.write(stream)
    stream.close()


try:
    opts, args = getopt(argv[1:], 'he:')
except GetoptError:
    print "Unrecognized argument(s) \'" + ", ".join(argv[1:]) + "\'. Make
sure that you\'ve entered an ein (-e).  " \
                                                          "Type
pyloader.py -h for help."
    exit(2)

for opt, arg in opts:
    if opt == '-e':
        ein = arg
    elif opt == '-h':
        print 'Command usage: extract_pdf_reader.py -e \"[ein]\"'
        exit(2)


# Ignore warnings.
filterwarnings("ignore")

#print "reader ein: " + ein
ein = '36-2545170'

# Get the FYE so we can download the PDF.
#conn = connect(host='localhost', user='root', password='rooster', db='irs')
conn = connect(host='localhost', user='root', password='mysql', db='irs')
cursor = conn.cursor()
cursor.execute('SELECT fye FROM irs_master WHERE ein = \'' + ein + '\'')
fye_send = cursor.fetchone()
print fye_send[0]
conn.close()

# Adjust ein.
ein = ein[0:2] + ein[3:len(ein)]
```

```
# Request the PDF.
get_pdf_link(ein, fye_send[0])
```

R Example

```
# NOTES

# NAs threw the original program, so this version relies on truncated
# data.  If I had more time, I would make a separate process for the
# NA section.

require(zoo);

# For this exercise, the following articles may be helpful:
#   https://www.clarusft.com/principal-component-analysis-of-the-swap-
curve-an-introduction/
#   https://doc.research-and-
analytics.csfb.com/docView?language=ENG&format=PDF&document_id=1001969
281&source_id=emcmt&serialid=Coz8ZUCgL92gmMydSBULHAsCBlmogDBprg0kSAhRL
Ck%3d
#   http://stats.stackexchange.com/questions/229092/how-to-reverse-
pca-and-reconstruct-original-variables-from-several-principal-com
#   http://stats.stackexchange.com/questions/57467/how-to-perform-
dimensionality-reduction-with-pca-in-r


# Load in swaps.csv as a zoo object (p)
p = read.zoo('swaps.csv', header = TRUE, index.column = 1, format =
"%Y-%m-%d", drop = FALSE, sep = ",");

# Only work with valid values (SEE NOTE AT BEGINNING OF DOCUMENT).
p = p[(max(which(is.na(p[,13])))+1):length(p[,1]),];



# Calculate the daily percent change for the data
swap_ret = coredata(p);
swap_helper = swap_ret;
n_cols = length(swap_ret[1,]);
length_cols = length(swap_ret[,1]);

for(i in 1:n_cols)
{
    for(j in 2:length_cols)
    {
        if(!(is.na(swap_helper[j-1,i])))
        {
            swap_ret[j,i] = (swap_helper[j,i]/swap_helper[j-1,i] -
1)*100;
        }
    }
}

# Returns are undefined for the first row.
swap_ret[1,] = 0;
```

```
# Find the prinicpal components of the swap data and
# plot the relative sizes of the prinicpal components (scree plot)
# Hint: prcomp()
n_X = length(p[1,]);
X = p[,1:n_X];

mu = colMeans(X);

Xpca = prcomp(X);

screeplot(Xpca);



# Identify how many principal components account for > 96% of the
variance (n)
# Hint: summary()
summary_helper = unname(unlist(summary(Xpca)[1]));

total_sd = sum(summary_helper);

bound = FALSE;
counter = 1;

while(identical(bound, FALSE))
{
    if(sum(summary_helper[1:counter])/total_sd > .96)
    {
        # Found how many components it took.
        counter = counter - 1;
        bound = TRUE;
        #print(counter);
    } else {
        counter = counter + 1;
    }
}



# For those n components, plot the coefficients (loadings) for each
swap, vs the swap maturity (yrs)
# Hint: Should be a plot with n lines, Coefficient Magnitude (y-axis)
vs Years (x-axis)

# Get the loadings.
loadings = list(list());

# Plotting bounds.
y_bottom = 0;
y_top = 0
```

```
for(i in 1:counter)
{
    loadings[[i]] = Xpca$rotation[,i];

    # Use the loop to get our plotting bounds.
    if(min(loadings[[i]] < y_bottom))
    {
        y_bottom = min(loadings[[i]]);
    }

    if(max(loadings[[i]] > y_top))
    {
        y_top = max(loadings[[i]]);
    }

}

# Get the years.
years = unlist(lapply(unlist(lapply(colnames(p), function(x)
strsplit(x, split='')), recursive = FALSE), function(y)
as.integer(paste(y[5:length(y)], collapse = ''))));

# Plot everything.
plot(years, loadings[[1]], col = 30, ylim = range(c(y_bottom,y_top)),
ylab = "Coefficient Magnitude", xlab = "Years");

if(counter > 1)
{
    for(i in 2:counter)
    {
        points(years, loadings[[i]], col = 30 + (i-1)*6);
    }
}




# Using the top n components,
#  (1) reconstruct (model) the one year swap returns (swap_ret_mod)
#  (2) Calculate the residuals between actual and modeled returns
(residuals_ret)
# swap_ret      : One year swap daily returns
# swap_ret_mod  : Model of one year swap daily returns
# residuals_ret : Error between actual returns and modeled returns
nComp = counter;
swap_ret_mod = Xpca$x[,1:nComp] %*% t(Xpca$rotation[,1:nComp]);
swap_ret_mod = scale(swap_ret_mod, center = -mu, scale = FALSE);

#swap_ret_mod = coredata(p);
residuals_ret = swap_ret;
swap_helper = swap_ret_mod;
```

```
for(i in 1:n_cols)
{
    for(j in 2:length_cols)
    {
        if(!(is.na(swap_helper[j-1,i])))
        {
            # (1)
            swap_ret_mod[j,i] = (swap_helper[j,i]/swap_helper[j-1,i] -
1)*100;
            # (2)
            residuals_ret[j,i] = swap_ret_mod[j,i] - swap_ret[j,i];
        }
    }
}

# Returns and residuals are undefined for the first row.
swap_ret_mod[1,] = 0;
residuals_ret[1,] = 0;


# Reconstruct the price series from daily returns, normalized to the
first value
#   in the series i.e. swap[1] = 1, swap_mod[1] = 1
# swap      : one year swap pricing
# swap_mod  : Model of one year swap prices
# residuals : Error between actual and modeled prices
swap = coredata(p);
swap[1,] = 1;

swap_mod = swap;

residuals = swap_mod;

for(i in 1:n_cols)
{
    for(j in 2:length_cols)
    {
        swap[j,i] = swap[j-1,i]*(1 + swap_ret[j]/100);
        swap_mod[j,i] = swap_mod[j-1,i]*(1 + swap_ret_mod[j]/100);
        residuals[j,i] = swap_mod[j,i] - swap[j,i];
    }
}



# Create a matrix of 2 charts on one plot (mfcol):
#   Top plot: Plot actual returns vs modeled returns
#   Bottom plot: Plot residual of actual returns vs modeled returns

# Knock off the first rows.
true_length = length(swap[,1]);
swap_ret = swap_ret[2:true_length,];
```

```
swap_ret_mod = swap_ret_mod[2:true_length,];
#residuals_ret = residuals_ret[2:true_length,];
swap = swap[2:true_length,];
swap_mod = swap_mod[2:true_length,];
residuals = residuals_ret[2:true_length,];

par(mfcol=c(2,1));

if(n_cols == 1)
{
    # If we have just one set, plot it.

    # See what our range is.
    y_max = max(swap_ret[,1], swap_ret_mod[,1]);
    y_min = min(swap_ret[,1], swap_ret_mod[,1]);

    plot(index(p)[2:length(index(p))], swap_ret[,1], pch = 1, col = 12,
ylim = c(y_min,y_max));
    points(index(p)[2:length(index(p))], swap_ret_mod[,1], pch = 2,
col = 18);
} else {

    # We have more than one set.

    # See what our range is.
    y_max = max(swap_ret, swap_ret_mod);
    y_min = min(swap_ret, swap_ret_mod);

    # Plot the first ones.
    plot(index(p)[2:length(index(p))], swap_ret[,1], pch = 1, col = 1,
ylim = c(y_min,y_max));
    points(index(p)[2:length(index(p))], swap_ret_mod[,1], pch = 2,
col = 8);

    for(i in 2:n_cols)
    {
        # Now the rest.
        points(index(p)[2:length(index(p))], swap_ret[,i], pch = i+1,
col = 6*i);
        points(index(p)[2:length(index(p))], swap_ret_mod[,i], pch =
i+2, col = 6*i);
    }
}

if(n_cols == 1)
{
    # If we have just one set, plot it.

    # See what our range is.
    y_max = max(residuals_ret[,1]);
    y_min = min(residuals_ret[,1]);
```

```
    plot(index(p), residuals_ret[,1], pch = 1, col = 12, ylim =
c(y_min,y_max));
} else {

    # We have more than one set.

    # See what our range is.
    y_max = max(residuals_ret);
    y_min = min(residuals_ret);

    # Plot the first ones.
    plot(index(p), residuals_ret[,1], pch = 1, col = 1, ylim =
c(y_min,y_max));

    for(i in 2:n_cols)
    {
        # Now the rest.
        points(index(p), residuals_ret[,i], pch = i+1, col = 6*i);
    }
}



# Create a matrix of 2 charts on one plot (mfcol):
#   Top plot: Plot actual prices vs modeled prices
#   Bottom plot: Plot residual of actual prices vs modeled prices
par(mfcol=c(2,1));

if(n_cols == 1)
{
    # If we have just one set, plot it.

    # See what our range is.
    y_max = max(swap[,1], swap_mod[,1]);
    y_min = min(swap[,1], swap_mod[,1]);

    plot(index(p)[2:length(index(p))], swap[,1], pch = 1, col = 12,
ylim = c(y_min,y_max));
    points(index(p)[2:length(index(p))], swap_mod[,1], pch = 2, col =
18);
} else {

    # We have more than one set.

    # See what our range is.
    y_max = max(swap, swap_mod);
    y_min = min(swap, swap_mod);

    # Plot the first ones.
    plot(index(p)[2:length(index(p))], swap[,1], pch = 1, col = 1,
ylim = c(y_min,y_max));
```

```
    points(index(p)[2:length(index(p))], swap_mod[,1], pch = 2, col =
8);

    for(i in 2:n_cols)
    {
        # Now the rest.
        points(index(p)[2:length(index(p))], swap[,i], pch = i+1, col
= 6*i);
        points(index(p)[2:length(index(p))], swap_mod[,i], pch = i+2,
col = 6*i);
    }
}

if(n_cols == 1)
{
    # If we have just one set, plot it.

    # See what our range is.
    y_max = max(residuals[,1]);
    y_min = min(residuals[,1]);

    plot(index(p)[2:length(index(p))], residuals[,1], pch = 1, col =
12, ylim = c(y_min,y_max));
} else {

    # We have more than one set.

    # See what our range is.
    y_max = max(residuals);
    y_min = min(residuals);

    # Plot the first ones.
    plot(index(p)[2:length(index(p))], residuals[,1], pch = 1, col = 1,
ylim = c(y_min,y_max));

    for(i in 2:n_cols)
    {
        # Now the rest.
        points(index(p)[2:length(index(p))], residuals[,i], pch = i+1,
col = 6*i);
    }
}
```

SQL Example

```
USE joyo_kanji;

-- Master List
SELECT * FROM joyo_kanji.master_list ORDER BY CAST(kanji_frequency_without_proper_nouns AS
UNSIGNED) DESC, number_of_on, number_of_meanings_of_on LIMIT 500;
SELECT strokes, kanji_frequency_without_proper_nouns FROM master_list;

SELECT * FROM master_list ORDER BY CAST(on_ratio_with_proper_nouns AS DECIMAL(5,5)) DESC;
SELECT on_ratio_without_proper_nouns FROM master_list ORDER BY on_ratio_without_proper_nouns
DESC, CAST(kanji_frequency_without_proper_nouns AS UNSIGNED) DESC;

SELECT * FROM master_list WHERE on_within_joyo LIKE '%[%';

SELECT number_of_on, number_of_meanings_of_on, COUNT(*) AS freq FROM master_list GROUP BY
number_of_on, number_of_meanings_of_on ORDER BY number_of_on, number_of_meanings_of_on;

SELECT * FROM joyo_kanji.master_list WHERE on_within_joyo LIKE '%koo%';

SELECT SUM(kanji_frequency_without_proper_nouns) FROM joyo_kanji.master_list;

SELECT SUM(CAST(kanji_frequency_without_proper_nouns AS UNSIGNED)) FROM
joyo_kanji.master_list;

DROP TABLE IF EXISTS kanji_classification_groupings;
CREATE TABLE kanji_classification_groupings AS (SELECT kanji_classification, COUNT(*) AS freq FROM
master_list GROUP BY kanji_classification ORDER BY freq DESC);
#UPDATE kanji_classification_groupings AS A LEFT JOIN (SELECT SUM(freq) AS rel FROM
kanji_classification_groupings) AS B ON A.freq != B.rel SET A.freq = 1;
#UPDATE kanji_classification_groupings SET rel = freq/2136 WHERE 1 = 1;
DESCRIBE master_list;


SELECT * FROM master_list WHERE kanji_classification = "国字 Original";

-- Who has only 1 on reading or 1 kun reading and then sort by number of meanings
SELECT * FROM master_list WHERE (number_of_on = 1 AND number_of_meanings_of_kun = 0) ORDER
BY number_of_meanings_of_on ASC;
SELECT * FROM master_list WHERE number_of_on = 0 AND
(number_of_kun_within_joyo_with_inflections = 1 AND
number_of_kun_within_joyo_without_inflections = 1) ORDER BY number_of_meanings_of_kun;

-- Most common readings within these
SELECT reading_within_joyo, COUNT(*) AS occurences FROM (SELECT * FROM master_list WHERE
(number_of_on = 1 AND number_of_meanings_of_kun = 0)) AS T GROUP BY reading_within_joyo
ORDER BY occurences DESC;

-- Sum of occurences
```

```
SELECT SUM(U.occurences) FROM (SELECT reading_within_joyo, COUNT(*) AS occurences FROM
(SELECT * FROM master_list WHERE (number_of_on = 1 AND number_of_meanings_of_kun = 0)) AS T
GROUP BY reading_within_joyo ORDER BY occurences DESC) AS U;


SELECT * FROM joyo_kanji.master_list;

SELECT * FROM joyo_kanji.master_list WHERE reading_within_joyo LIKE '%],[%';

SELECT * FROM joyo_kanji.master_list ORDER BY number_of_meanings_of_on DESC, number_of_on
DESC;

SELECT COUNT(*) FROM joyo_kanji.master_list WHERE number_of_on = '1';


-- Other Lists

SELECT number_of_common_on_within_joyo, COUNT(*) FROM joyo_kanji.master_list_with_splitout
GROUP BY number_of_common_on_within_joyo;

SELECT * FROM joyo_kanji.master_list_with_splitout ORDER BY CAST(radical_freq AS UNSIGNED) DESC;


SELECT jlpt_test, COUNT(*) FROM joyo_kanji.master_list_with_splitout GROUP BY jlpt_test;
SELECT COUNT(*) FROM joyo_kanji.master_list_with_splitout WHERE reading_beyond_joyo != '';

UPDATE joyo_kanji.master_list_with_splitout SET reading_beyond_joyo = '-' WHERE
reading_beyond_joyo = '';

SELECT * FROM joyo_kanji.master_list_with_splitout WHERE on_within_joyo LIKE 'くわ-える';

SELECT * FROM joyo_kanji.master_list_with_splitout WHERE number_of_on > 0 ORDER BY
CAST(number_of_on AS UNSIGNED) ASC, CAST(on_ratio_with_proper_nouns AS DECIMAL(5,5)) DESC,
CAST(kanji_frequency_with_proper_nouns AS UNSIGNED) DESC;

-- most of these have only an on reading.  unfortunately they only represent ~ 2.5% of all joyo kanji
SELECT * FROM master_list_with_splitout WHERE max_readings = 1 AND (number_of_meanings_of_on
= 1 OR number_of_meanings_of_kun = 1);

-- produces some errors
SELECT temp, COUNT(*) AS freq FROM joyo_kanji.common_on_readings GROUP BY temp;

kanji id 1334, incorrect number of on readings

SELECT name_of_radical, COUNT(*) AS freq FROM joyo_kanji.master_list_with_splitout GROUP BY
name_of_radical ORDER BY freq DESC;
```

```
SELECT * FROM joyo_kanji.master_list_with_splitout ORDER BY max_readings ASC,
number_of_meanings_of_on ASC, number_of_meanings_of_kun ASC;

DESCRIBE joyo_kanji.master_list_with_splitout;

SELECT max_readings, COUNT(*) FROM joyo_kanji.master_list_with_splitout WHERE
CAST(cumulative_frequency_with_proper_nouns AS UNSIGNED) <= .51 GROUP BY max_readings;



SELECT on_readings_common_only, COUNT(*) AS freq FROM joyo_kanji.on_readings_common_only
GROUP BY on_readings_common_only ORDER BY freq DESC;

SELECT * FROM joyo_kanji.on_readings_common_only;

SELECT * FROM joyo_kanji.on_readings_common_and_uncommon;
SELECT on_readings_common_and_uncommon, COUNT(*) AS freq FROM
joyo_kanji.on_readings_common_and_uncommon GROUP BY on_readings_common_and_uncommon
ORDER BY freq DESC;
```