**Manual flow cytometry analysis  
Post-session activities**

1. **Loading packages and reading in data**
   1. Load the following packages:
      1. flowCore (for reading .fcs files to flowFrames/flowSets)
      2. ggcyto (for plotting flowFrames/flowSets)
      3. flowCut (for cleaning flowFrames)
      4. flowWorkspace (for creating gatingSets and adding gates to them)
   2. Use the read.flowSet() function to read in either:
      1. your own experiment of fcs files (with their own spillover matrix from the instrument)  
         OR
      2. the example dataset we used in the tutorial
   3. Use the fsApply() function to find the number of events and columns in each flowFrame in the flowSet
2. **Cleaning, compensating, and transforming the flowSet**
   1. Use the fsApply() function to apply the flowCut algorithm to each flowFrame in the flowSet, and return a flowSet of cleaned data. Store this new flowSet in an object  
      **Hint:** In the handout, I show you a slightly different way of doing this by creating an *anonymous function*. We didn’t cover this in the sessions, but it is a way of defining a function on-the-fly without storing it first. Let me know if you would like more explanation of this.
   2. Check the new flowCut directory in your working directory. Take a look at the images (if any) that have been created in this directory and interpret the plots
   3. Extract the acquisition-defined spillover matrix from the appropriate keyword and compensate the flowSet
   4. Define the parameters that should be transformed, and apply a logicle transformation to them  
      **Hint:** use the estimateLogicle() function on a single flowFrame to define the transformations, then use the transform() function to apply them to the flowSet
3. **Plotting the data**
   1. Use the ggcyto() function to plot some parameters of interest against each other. Experiment with different values of the bins argument for geom\_hex()
4. **Gating the data**
   1. Create a gatingSet for the flowSet
   2. Gate the data as you please! If you are working on your own data, try to replicate any manual analysis you may have done on it in FlowJo
   3. **Hint:** remember the general process for getting the data is:  
      - plot the parameters you wish to gate on  
      - define the gate using one of the gating functions   
       (rectangleGate(), polygonGate(), ellipsoidGate() etc)  
      - add the gate to the gatingSet using gs\_pop\_add()  
      - recompute() the population memberships  
      - visualise the gates with autoplot()  
      - if you are not happy with the gate, remove it with gs\_pop\_remove(), change the gate, and re-add it (see Section 5.3 of the handout),  
      - use gs\_pop\_get\_data() to extract the gated events into a new flowSet so you can plot this data ready to make the next gate
   4. Call plot() on your gating set to visualise your gating hierarchy
   5. Call autoplot() on an individual flowFrame from your gatingSet   
      **Hint:** if your gatingSet is called gs, try autoplot(gs[[1]])
5. **Extracting population statistics**
   1. Use the gs\_pop\_get\_stats() function to create a table of summary statistics for counts and percentages for all populations, and a separate table of MFI values for only your terminal populations of interest  
      **Hint:** use the nodes argument to specify which populations you want
   2. Save these summary tables as .csv files using the write.csv() function
6. **Export cleaned data**
   1. Use the gs\_pop\_get\_data() function to extract a flowSet containing only a “clean” population of interest such as all singlets, or all CD45
   2. Use the write.flowSet() function to write out new .fcs files that contain only this “clean” population (see section 9 of the handout)

**If you have any questions or want feedback, please contact** [**hefin.rhys@ucb.com**](mailto:hefin.rhys@ucb.com)