/\* **2. The Gene Ontology (GO)-derived protein complexes (PC)** \*/

/\* **2.1 Identify the PC** \*/

/\* The second aim is to retrieve all the PC reported by GO (in the CC sub-ontology). Start by downloading go-basic.obo from geneontology.org. Make the necessary changes to bring the large file into the desired format. \*/

.open c:/sqlite/PC/pc.db

.mode tabs

.headers ON

.import c:/sqlite/PC/inputs/go\_basic.txt tmp1\_go

CREATE TABLE tmp2\_go(description TEXT);

INSERT INTO tmp2\_go

SELECT \* FROM tmp1\_go

WHERE (

SUBSTR(description,1,6) = ‘[Term]’ OR SUBSTR(description,1,3) = ‘id:’ OR SUBSTR(description,1,5) = ‘name:’ OR SUBSTR(description,1,10) = ‘namespace:’ OR SUBSTR(description,1,5) = ‘is\_a:’ OR SUBSTR(description,1,13) = ‘relationship:’

);

DROP TABLE tmp1\_go;

/\* Add a row id column and then export the tmp2\_go.txt table in txt format. \*/

ALTER TABLE tmp2\_go ADD COLUMN row\_counter INTEGER;

UPDATE tmp2\_go SET row\_counter = ROWID;

.once c:/sqlite/PC/inputs/tmp2\_go.txt

SELECT \* FROM tmp2\_go;

/\* Elaborate the file in Excel, to attribute an id to all the different attributes (e.g., id, name, namespace etc) that refer to the same item (i.e., the same GO term). Save it as go\_basic2.txt. Import it in the SQL database as tmp3\_go. Finally, in each tmp4\_go\_[field] table, add one by one the various attributes (id, name, namespace, parents and related terms). \*/

.import c:/sqlite/PC/inputs/go\_basic2.txt tmp3\_go

CREATE TABLE tmp4\_go\_id(term INTEGER, go\_id TEXT); /\* The GO identifier \*/

INSERT INTO tmp4\_go\_id

SELECT DISTINCT \* FROM(

SELECT term, text FROM tmp3\_go WHERE field = ‘id’);

CREATE TABLE tmp4\_go\_name(term INTEGER, go\_name TEXT); /\* The GO name \*/

INSERT INTO tmp4\_go\_name

SELECT DISTINCT \* FROM(

SELECT term, text FROM tmp3\_go WHERE field = ‘name’);

CREATE TABLE tmp4\_go\_namespace(term INTEGER, go\_namespace TEXT); /\*The sub-ontology \*/

INSERT INTO tmp4\_go\_namespace

SELECT DISTINCT \* FROM(

SELECT term, text FROM tmp3\_go WHERE field = ‘namespace’);

CREATE TABLE tmp4\_go\_parents(term INTEGER, parent\_id TEXT); /\* The GO parent(s) \*/

INSERT INTO tmp4\_go\_parents

SELECT DISTINCT \* FROM(

SELECT term, text FROM tmp3\_go WHERE field = ‘is\_a’);

ALTER TABLE tmp4\_go\_parents ADD COLUMN link TEXT;

UPDATE tmp4\_go\_parents SET link = ‘is\_a’;

CREATE TABLE tmp4\_go\_relations(term INTEGER, relationship TEXT); /\* The related terms \*/

INSERT INTO tmp4\_go\_relations

SELECT DISTINCT \* FROM(

SELECT term, text FROM tmp3\_go WHERE field = ‘relationship’);

ALTER TABLE tmp4\_go\_relations ADD COLUMN related\_id TEXT;

UPDATE tmp4\_go\_relations SET related\_id = SUBSTR(relationship,-10,10);

ALTER TABLE tmp4\_go\_relations ADD COLUMN link TEXT;

UPDATE tmp4\_go\_relations SET link = SUBSTR(relationship,1,LENGTH(relationship)-11);

/\* Merge the parents and the related terms in one table. \*/

CREATE TABLE tmp4\_go\_links(term INTEGER, link TEXT, linked\_id TEXT);

INSERT INTO tmp4\_go\_links

SELECT DISTINCT \* FROM(

SELECT term, link, parent\_id FROM tmp4\_go\_parents

UNION

SELECT term, link, related\_id FROM tmp4\_go\_relations);

CREATE TABLE tmp\_go\_basic\_def(go\_id TEXT, go\_name TEXT, space TEXT, link TEXT, linked\_id TEXT);

INSERT INTO tmp\_go\_basic\_def

SELECT DISTINCT \* FROM(

SELECT tmp4\_go\_id.go\_id, tmp4\_go\_name.go\_name, tmp4\_go\_namespace.go\_namespace, tmp4\_go\_links.link, tmp4\_go\_links.linked\_id

FROM tmp4\_go\_id

LEFT JOIN tmp4\_go\_name ON tmp4\_go\_id.term = tmp4\_go\_name.term

LEFT JOIN tmp4\_go\_namespace ON tmp4\_go\_id.term = tmp4\_go\_namespace.term

LEFT JOIN tmp4\_go\_links ON tmp4\_go\_id.term = tmp4\_go\_links.term

);

/\* Add the name of the linked term (note that the procedure deletes many parent-less terms, namely the obsolete terms and the three root terms (‘BP’, ‘CC’ and ‘MF’). \*/

CREATE TABLE **go\_basic\_def**(go\_id TEXT, go\_name TEXT, space TEXT, link TEXT, linked\_id TEXT, linked\_name TEXT);

INSERT INTO go\_basic\_def

SELECT DISTINCT \* FROM(

SELECT i.go\_id AS ‘go\_id’, i.go\_name AS ‘go\_name’, i.space AS ‘space’, i.link AS ‘link’, i.linked\_id AS ‘linked\_id’, j.go\_name AS ‘linked\_name’

FROM tmp\_go\_basic\_def i

INNER JOIN tmp\_go\_basic\_def j

ON i.linked\_id = j.go\_id);

.once c:/sqlite/PC/inputs/go\_basic\_def.txt

SELECT \* FROM go\_basic\_def;

DROP TABLE tmp3\_go; DROP TABLE tmp4\_go\_id;

DROP TABLE tmp4\_go\_name; DROP TABLE tmp4\_go\_namespace;

DROP TABLE tmp4\_go\_parents; DROP TABLE tmp4\_go\_relations; DROP TABLE tmp4\_go\_links;

DROP TABLE tmp\_go\_basic\_def;

/\* Finally, from the go\_basic\_def table retrieve the presumptive PC. \*/

CREATE TABLE tmp1\_pc(go\_id TEXT, go\_name TEXT, space TEXT, link TEXT, linked\_id TEXT, linked\_name TEXT);

INSERT INTO tmp1\_pc

SELECT DISTINCT \* FROM(

SELECT \* FROM go\_basic\_def

WHERE space = ‘CC’ AND go\_name LIKE ‘%complex%’);

/\* There are 1,847 distinct go\_id, of which 589 are also go\_parent\_id (or object of a part\_of relation). Identify the terms that are (‘P’) and that are not parents (‘NP’) of other terms. \*/

CREATE TABLE tmp\_linked(go\_id TEXT);

INSERT INTO tmp\_linked

SELECT DISTINCT \* FROM(SELECT linked\_id FROM tmp1\_pc);

CREATE TABLE **pc**(go\_id TEXT, go\_name TEXT, status TEXT, link TEXT, linked\_id TEXT, linked\_name TEXT);

INSERT INTO pc

SELECT DISTINCT \* FROM(

SELECT tmp1\_pc.go\_id, tmp1\_pc.go\_name, tmp\_linked.go\_id, tmp1\_pc.link, tmp1\_pc.linked\_id, tmp1\_pc.linked\_name

FROM tmp1\_pc

LEFT JOIN tmp\_linked

ON tmp1\_pc.go\_id = tmp\_linked.go\_id);

UPDATE pc SET status = (CASE WHEN (status IS NULL) THEN ‘NP’ ELSE ‘P’ END);

.once c:/sqlite/PC/outputs/pc.txt

SELECT \* FROM pc;

DROP TABLE tmp1\_pc;

DROP TABLE tmp\_linked;

/\* **2.2 Identify the PC-associated proteins** \*/

/\* Retrieve all the proteins annotated in GOA within these complexes. \*/

/\* I did not download the GOA file from the GO or the EBI ftp sites, because they enlist the genes as UniProtKB id. Rather, I downloaded the gene2go.gz file from Entrez (ftp://ftp.ncbi.nih.gov/gene/DATA/), which is updated daily. I prepared a smaller file, just searching for human genes (taxon\_id = 9606) and removing the relatively few entries in which a gene is not (qualifier ‘NOT’) associated with a GO term. Also, make few changes (replace ‘GO:’ with ‘GO\_’, set ‘BP/CC/MF’ instead of ‘Process/Component/Function’). Export the smaller table as gene2go.txt. \*/

.import c:/SQLITE/PC/inputs/gene2go.txt tmp1\_gene2go

/\* First, the gene annotations with GO terms in the CC sub-ontology. \*/

CREATE TABLE tmp2\_gene2go(gene\_id TEXT, go\_id TEXT);

INSERT INTO tmp2\_gene2go

SELECT DISTINCT \* FROM(

SELECT gene\_id, go\_id

FROM tmp1\_gene2go WHERE space = ‘CC’);

.import c:/sqlite/PC/inputs/entrez\_genes.txt entrez

CREATE TABLE **gene2go**(gene\_id TEXT, symbol TEXT, go\_id TEXT);

INSERT INTO gene2go

SELECT DISTINCT \* FROM(

SELECT tmp2\_gene2go.gene\_id, entrez.Symbol, tmp2\_gene2go.go\_id

FROM tmp2\_gene2go

LEFT JOIN entrez ON tmp2\_gene2go.gene\_id = entrez.GeneId);

DROP TABLE tmp1\_gene2go;

DROP TABLE tmp2\_gene2go;

CREATE TABLE **pc\_prot**(go\_id TEXT, go\_name TEXT, status TEXT, gene\_id INTEGER, symbol TEXT);

INSERT INTO pc\_prot

SELECT DISTINCT \* FROM(

SELECT pc.go\_id, pc.go\_name, pc.status, gene2go.gene\_id, gene2go.symbol

FROM pc

JOIN gene2go

ON pc.go\_id = gene2go.go\_id);

/\* Calculate the PC length (i.e., the number of proteins they contain). \*/

CREATE TABLE **pc\_length**(go\_id TEXT, go\_name TEXT, status TEXT, pc\_length INTEGER);

INSERT INTO pc\_length

SELECT DISTINCT \* FROM(

SELECT go\_id, go\_name, status, COUNT (DISTINCT gene\_id) FROM pc\_prot

GROUP BY go\_id, go\_name, status

ORDER BY COUNT (DISTINCT gene\_id) DESC);

/\* Finally, define the PC of interest as the PC that contain at least two proteins. Then, delete the three generic PC (i.e., Protein complex GO:0043234, Transcription factor complex GO:0005667 and Macromolecular complex GO:0032991). \*/

CREATE TABLE **pc\_prot\_def**(go\_id TEXT, go\_name TEXT, status TEXT, gene\_id INTEGER, symbol TEXT, pc\_length INTEGER);

INSERT INTO pc\_prot\_def

SELECT DISTINCT \* FROM(

SELECT a.go\_id, a.go\_name, a.status, a.gene\_id, a.symbol, b.pc\_length

FROM pc\_prot a

JOIN pc\_length b ON (a.go\_id = b.go\_id AND b.pc\_length > 1)

);

DELETE FROM pc\_prot\_def WHERE(go\_id = ‘GO\_0043234’ OR go\_id = ‘GO\_0005667’ OR go\_id = ‘GO\_0032991’);