المحتويات

مقدمة *Introduction* **vii**

٠.١ الايجابيات والسلبيات للغة ال آر *Pros and Cons of R* x

٠.١.١ مزايا اللغة *Language Pros* x

٠.١.٢ عيوب اللغة *Language Cons* xiv

تمهيد *Preface* **xvii**

شكر وعرفان *Acknowledgment* **xix**

حول المؤلف *About the Author* **xxi**

١ البداية مع ال آر *Start with R* **١**

١.١ تحميل ال آر *R download* ٢

١.٢ مقارنة بين $32-bit$ و $64-bit$ ٣

١.٣ التنصيب *Installation* ٣

١.٣.١ التنصيب على نظام الوندوز ٣

١.٣.٢ التنصيب على نظام الماك ١٣

٢ بيئة ال آر *R Environment* **١٥**

٢.١ واجهة الأوامر *Command Interface* ١٦

٢.٢ بيئة التطوير المتكاملة *IDE* ١٦

٢.٣ آر ستديو *RStudio* ١٧

٢.٤ مشاريع آر ستوديو *RStudio Projects* ١٩

٢.٥ أدوات آر ستوديو *RStudio Tools* ٢١

٢.٦ الكتابة في وحدة التحكم *Writing in Console* ٢٢
٢.٧ محرر الكود *Code Editor* ٢٣
٢.٨ التحديث *Updating* ٢٥
٢.٨.١ تحديث ال آر *R Updating* ٢٥
٢.٨.٢ تحديث ال آر ستديو *RStudio Updating* ٢٧

٣ حزم ال آر *R Packages* ٢٩
٣.١ الحزم الأساسية *Standard Packages* ٢٩
٣.٢ الحزم الخارجية *External Packages* ٣٠
٣.٢.١ المستودعات *Repositories* ٣٠
٣.٣ اوامر مفيدة للحزم *Useful Commands for Packages* ٣٤
٣.٣.١ تنصيب حزم متعددة *Install Multiple Packages* ٣٤
٣.٣.٢ تحديث الحزم *Packages Updating* ٣٤
٣.٣.٣ مصدر الكود للدالة *Function Source Code* ٣٥
٣.٣.٤ التحقق من الحزم المثبتة *Check for Installed Packages* . . ٣٦
٣.٣.٥ الدوال المصاحبة للحزم *Functions Accompanying Packages* ٣٦
٣.٣.٦ نصائح مفيدة لتثبيت الحزم *Helpful Tips for Package Installations* ٣٧
٣.٤ المساعدة لفهم عمل الدالة ٣٧

٤ أساسيات ال آر *Basics of R* ٤١
٤.١ المتغيرات *Variables* ٤١
٤.١.١ انواع البيانات *Data Types* ٤٢
٤.٢ تركيب البيانات *Data Stracture* ٤٩
٤.٢.١ المتجهات *Vectors* ٥٠
٤.٢.٢ العمليات الحسابية على المتجهات ٥٥
٤.٢.٣ توليد تسلسلات منتظمة ٥٦
٤.٢.٤ تعديل وتصحيح البيانات ضمن نطاق معين ٥٨
٤.٢.٥ المصفوفة *Matrix* ٦٠
٤.٢.٦ المجموعة *Array* ٧٢
٤.٢.٧ البيانات المؤطرة *Data Frame* ٧٨
٤.٢.٨ تبليز *Tibbles* ٨٦
٤.٢.٩ القوائم *Lists* ٨٩
٤.٢.١٠ العوامل *Factors* ٩٩

٤.٣ القيم المفقودة ١٠٨

٥ المشغلات في ال آر *R Operators* **١١١**
٥.١ المشغلات الحسابية *Arithmetic Operators* ١١٢
٥.٢ المشغلات الترابطية *Relational Operators* ١١٣
٥.٣ المشغلات المنطقية *Logical Operators* ١١٤
٥.٤ المشغلات التخصصية *Assignment Operators* ١١٧
٥.٥ المشغلات المتفرقة *Miscellaneous Operators* ١١٩
٥.٦ مشغل العبور آر *Pipe Operator R* ١٢١
٥.٦.١ الاستخدام الاساسي لمشغلات العبور ١٢٢
٥.٦.٢ الدوال التسلسلية ١٢٢
٥.٦.٣ التخصيص مع %<>% ١٢٤
٥.٦.٤ عرض المحتويات مع %$% ١٢٥
٥.٦.٥ خلق بيانات جانبية مع %T>% ١٢٦

٦ قراءة البيانات *Reading Data* **١٢٩**
٦.١ ضبط دليل العمل *Settingthe Working Directory* ١٢٩
٦.٢ قراءة ملفات ال *CSV* ١٣٠
٦.٢.١ الادخال كملف *CSV* ١٣٠
٦.٢.٢ قراءة ملف ال *CSV* ١٣١
٦.٢.٣ تحليل البيانات في ملف ال *CSV* ١٣٣
٦.٢.٤ الكتابة في ملف ال *CSV* ١٣٥
٦.٣ قراءة ملفات ال *Excel* ١٣٦
٦.٤ قراءة ملفات ال *TXT* ١٣٩
٦.٥ قراءة ملفات ال *XML* ١٤٣
٦.٦ قراءة ملفات ال *JSON* ١٤٧
٦.٧ قراءة قواعد البيانات *Reading Databases* ١٥٠
٦.٧.١ بنية قاعدة البيانات ١٥٠
٦.٧.٢ صعوبات العمل مع مجموعات البيانات الكبيرة ١٥١
٦.٧.٣ استخدام *dplyr* للاستعلام عن قاعدة البيانات ١٥١
٦.٨ بيانات من ادوات احصائية اخرى ١٥٥
٦.٩ ملفات ال *R* الثنائية *Binary Files* ١٥٦
٦.١٠ بيانات ضمن ال *R* ١٥٩

٦.١١ استخلاص البيانات من المواقع ١٦٠
٦.١١.١ جداول ال $HTML$ البسيطة ١٦٠
٦.١١.٢ تشذيب بيانات الانترنت ١٦١

٧ رسومات أحصائية $Statistical\ Graphics$ **١٦٣**
٧.١ الرسومات الأساسية $Base\ Graphics$ ١٦٣
٧.١.١ الرسوم البيانية الأساسية $Base\ Histograms$ ١٦٤
٧.١.٢ الرسوم المبعثرة الاساسي $Base\ Scatterplot$ ١٦٥
٧.١.٣ الرسم الصندوقي $Boxplot$ ١٦٦
٧.١.٤ الرسم الدائري $Pie\ Chart$ ١٦٧
٧.١.٥ الرسم الشريطي $Bar\ Chart$ ١٧٠
٧.٢ حزمة ال $ggplot2$ ١٧٢
٧.٢.١ الرسوم البيانية والكثافات $Histogram\ and\ Densities$ ١٧٣
٧.٢.٢ الرسومات المبعثرة $Scotterplot$ ١٧٤
٧.٢.٣ الرسم الصندوقي والكماني $Boxplot\ and\ Violon\ plot$ ١٧٩
٧.٢.٤ الرسم الخطي $Line\ Graph$ ١٨٢
٧.٢.٥ دوال ال $themes$ ١٨٥

٨ عبارات التحكم $Control\ Statements$ **١٨٧**
٨.١ عبارة أذا $if\ Statement$ ١٨٧
٨.٢ عبارة $if\ and\ else$ ١٩٠
٨.٢.١ شروط متداخلة $Nested\ Conditions$ ١٩٢
٨.٢.٢ عبارات متعددة $Multiply\ Statements$ ١٩٣
٨.٢.٣ أرجاع القيم $Return\ Values$ ١٩٥
٨.٢.٤ فحوصات السلامة $Sanity\ checks$ ١٩٥
٨.٣ عبارة $Switch$ ١٩٨
٨.٤ عبارة $ifelse$ ١٩٩
٨.٤.١ الاختبارات المركبة $Compound\ Tests$ ٢٠١

٩ الحلقات $Loops$ **٢٠٣**
٩.١ $for\ loop$ ٢٠٣
٩.٢ $while\ loop$ ٢١٥
٩.٣ $repeat\ loop$ ٢١٧
٩.٤ $next\ and\ break\ loop$ ٢١٨

٩.٥ النتائج المرحلية *Interim Results* ٢٢٠
٩.٥.١ *Fixed* ٢٢٠
٩.٥.٢ *Dynamic* ٢٢٣
٩.٦ *Vectorization in R* ٢٢٥

١٠ كتابة دوال *Writing Functions* **٢٢٧**
١٠.١ بناء الدالة *Function Construct* ٢٢٩
١٠.١.١ الدالة الداخلية *Built – in Function* ٢٣١
١٠.١.٢ الدالة المعرفة *User – defined Function* ٢٣١
١٠.٢ مدخلات الدالة *Function Arguments* ٢٣٢
١٠.٢.١ المدخلات الأساسية *Default Arguments* ٢٣٤
١٠.٢.٢ مدخلات زائدة *Extra Arguments* ٢٣٤
١٠.٢.٣ استدعاء الدالة *Function Calling* ٢٣٥
١٠.٣ أعادة القيم *Return Values* ٢٣٧

١١ التلاعب الجماعي *Group Manipulation* **٢٤١**
١١.١ *Apply Family* ٢٤٢
١١.١.١ *apply* ٢٤٢
١١.١.٢ *lapply and sapply* ٢٤٣
١١.١.٣ *mapply* ٢٤٥
١١.١.٤ *Other apply Functions* ٢٤٥
١١.٢ *aggregate* ٢٤٦
١١.٣ *plyr* ٢٤٨
١١.٣.١ *ddply* ٢٤٩
١١.٣.٢ *llply* ٢٤٩
١١.٣.٣ *plyr Helper Functions* ٢٥١
١١.٣.٤ *Speed versus Convenience* ٢٥١
١١.٤ البيانات الجدولية *data.table* ٢٥١
١١.٤.١ *Keys* ٢٥٣

١٢ إعادة تشكيل البيانات *Data Reshaping* **٢٥٧**
١٢.١ توحيد الأعمدة والصفوف *cbind and rbind* ٢٥٩
١٢.٢ تبديل موضع المصفوفة *Transpose a matrix* ٢٦٠
١٢.٣ الانضمام *Join* ٢٦١

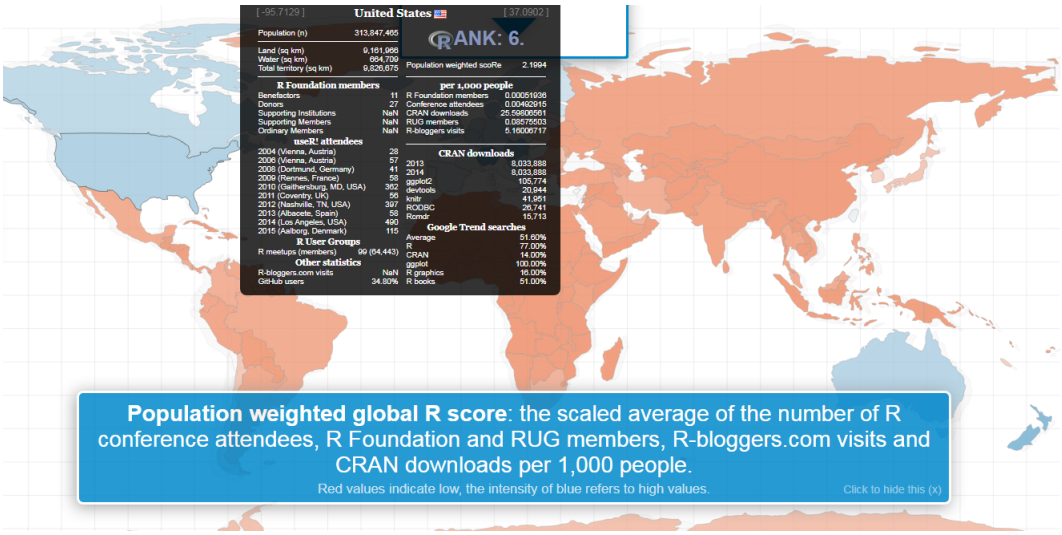
١٢.٣.١ الاندماج *merge* ٢٦١

١٣ التلاعب بالنصوص *Manipulating Strings* **٢٦٩**
١٣.١ دوال النصوص *String Functions* ٢٦٩
١٣.١.١ *grep* ٢٧٠
١٣.١.٢ *nchar* ٢٧١
١٣.١.٣ *paste* ٢٧٢
١٣.١.٤ *sprintf* ٢٧٣
١٣.١.٥ *substr* ٢٧٤
١٣.١.٦ *strsplit* ٢٧٥
١٣.١.٧ *regexpr* ٢٧٥
١٣.٢ استخراج النص *Extracting Text* ٢٧٦
١٣.٣ تعابير *Expressions* ٢٨١

مقدمة *Introduction*

إن أصدار هذا الكتاب (كخطوة أولى) ناتج عن الحاجة الماسة لسد الفجوة الموجودة في اللغات الإحصائية في المناهج العربية. فمن المهم جداً عند الولوج إلى علم الإحصاء من أجل التحاليل الإحصائية للتجارب الميدانية أن يتم فهم عمل المعادلات الإحصائية وكيفية معالجة البيانات كخطوة أولى لمخرجات رصينة. إذ يُدرس في أغلب الجامعات والمعاهد علم الإحصاء كأحد الطرق العلمية لجمع البيانات وتنظيمها وتحليلها للوصول إلى نتائج موثوقة تُسهم في دعم القرارات المستقبلية. وإن اتساع إستخدام علم الإحصاء ناتج عن التطورات السريعة في العلوم المختلفة وكمية البيانات العملاقة التي من الممكن الحصول عليها في زمن وجهد قياسيين. هذا الإتساع كان لابد أن يؤدي إلى تحفيز المختصين على إنشاء أو تبني بعض الأدوات التي تُسهم في تسهيل عملية التنظيم والتحليل وغيرها للنتائج. على أثر ذلك انتشرت البرامج الإحصائية المختلفة, فقد أصبحت البرامج الإحصائية أكثر تداولاً بل تُدرس في مختلف المعاهد والجامعات المنتشرة وفي تطبيقات الأعمال الهامة ، وهذا ما نلمسه واضحا في تدريس برنامج معالج الجداول Excel والبرنامج الإحصائي SPSS وبرنامج Matlab و Minitab وبرنامج S و SAS .

تُعد لغة ال R الإحصائية من أكثر اللغات انتشاراً واهمها. إذ من الممكن معرفة عدد المستخدمين الذي بلغ ملايين حول العالم من خلال بحث ذلك في محرك البحث $google$. بالرغم من العدد الهائل من المستخدمين إلا اننا نعتمد العلمية أكثر في طرح الموضوع من خلال معرفة أين يكمن النشاط الأكثر (إستخدام لغة ال R) بين دول العالم. معرفة تمركز النشاط يعطي دلالة واضحة حول أهمية الدول المستخدمة لهذه اللغة على المستوى التعليمي, الإقتصادي , المالي, الزراعي وغيرها. للإجابة على هذا السؤال قام فريق من مستخدمي R في $Rapporter$ بدمج البيانات حول مواقع أعضاء مؤسسة الR ومؤلفي الحزم $packages$ وتنزيل الحزم وأعضاء مجموعة المستخدمين في "درجة إستخدام الR" واحدة على مستوى الدولة. قاموا بعد ذلك بتعديل نتائج مستخدم الR هذه لتناسب عدد السكان وعرضها على الخريطة أدناه: يمثل اللون الأحمر أقل نشاط في إستخدام الR, بينما يمثل اللون الأزرق الأكثر نشاطاً في إستخدام ال R. الخريطة الاتية توضح الأعداد والأرقام للولايات المتحدة الامريكية.

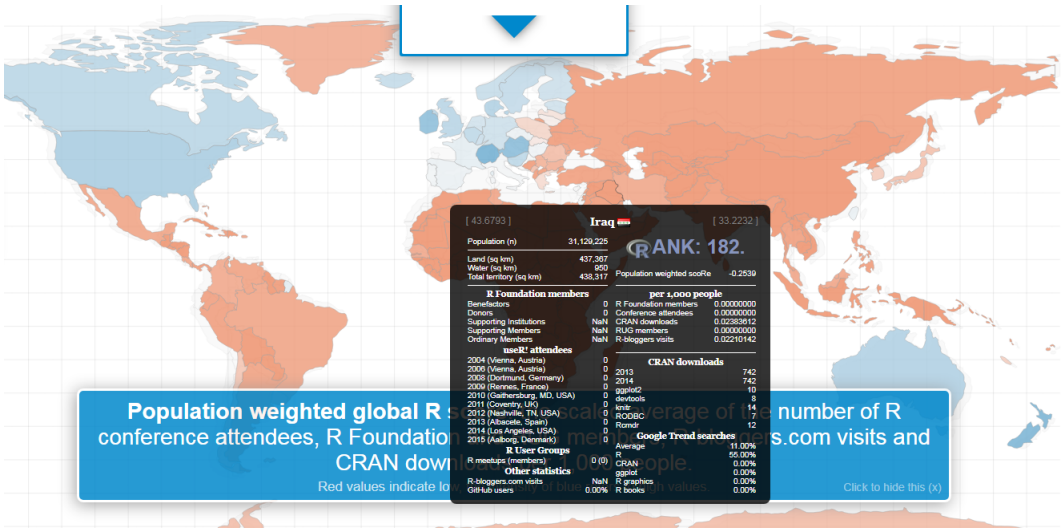


شكل ١: خارطة توضح توزيع اماكن وكثافة إستخدام لغة ال R حول العالم (الولايات المتحدة مثال)

من خلال المعطيات المقدمة من قبل فريق مستخدمي الR في *Rapporter* يمكن تحديد الدول الأكثر نشاطاً في إستخدام لغة ال R على النحو الآتي:

- سويسرا *Switzerland*.
- نيوزلندا *New Zealand (birthplace of R)*.
- النمسا *Austria (home of the R Foundation)*.
- ارلندا *Ireland*.
- جزر الولايات المتحدة البعيدة الصغرى *United States Minor Outlying Islands*.
- الولايات المتحدة *United States*.
- استراليا *Australia*.
- سنغابورا *Singapore*.
- الدنمارك *Denmark*.
- المملكة المتحدة *UnitedKingdom*.
- كندا *Canada*.

من خلال ما تم إدراجة من أسماء دول تُعد رائدة في المجالات العلمية والأدبية المختلفة, يتضح لنا دور لغة ال R الهام جداً في كثير من الأمور الإحصائية والرسوم البيانية وغيرها. نود أن نشير هنا إلى دور العراق أيضاً في هذه المسألة إذ تم تأليف هذا الكتاب لدفع بلدنا العزيز نحو مصاف الدول المتقدمة. توضح الخارطة الاتية الحاجة الملحة لإدخال لغة ال R إلى الوسط الأكاديمي, المهني والتطبيقي في العراق. إذ تشير البيانات إلى أن هناك مستوى شبه معدوم في إستخدام لغة ال R في العراق علماً أن البيانات لهذه الخارطة تعود لعام 2014.



شكل ٢: خارطة توضح توزيع اماكن وكثافة إستخدام لغة ال R حول العالم (العراق مثال)

ما لا يخفى على الكثير من العاملين في مجال البرمجة أن لغة ال R تستمد قوتها من الكثير من الحزم $packages$ الموجودة وبشكل مجاني تكاد تكون بالالاف وهذا ما يحتاجه المواطن في أغلب بلدان العالم الثالث (العلم للجميع). هذه الحزم تعمل على اعطاء الكثير من التحاليل الإحصائية وكذلك الرسومات البيانية (من البسيطة إلى المتطورة جدا) التي تكون لها ضرورة عند عرض العمل في الورش, الندوات, المؤتمرات وكذلك البحوث.

بالرغم من الامكانية الهائلة التي تقدمها هذه اللغة, وانتشارها الواسع بين دول العالم الأول المتطورة, إلا إنها لا تخلو من بعض العيوب. يجب أن لا يفهم من كلمة العيوب أن اللغة تحتوي على مشاكل لا تعاني منها بقية لغات البرمجة, انما لكل لغة عيوبها الخاصة بها. من أجل الوقوف على أهمية اللغة وإختيارها كلغة تستحق بذل المجهود والوقت معها, وجب الوقوف عند أهم المزايا والعيوب لديها. من هنا يمكن رؤية فيما اذا كانت المزايا تتفوق على العيوب أو لا.

About the Author حول المؤلف

Dr.AbdulkareemSahabAldabsaa د. عبدالكريم سحاب الدبسا

حصل المؤلف على شهادة الدكتوراه في مجال الموارد المائية وأدارة الترب الزراعية (تتبع حركة المياه في أسطح الترب وبين المسامات وتحديد صفات التربة) من كلية الزراعة والبيئة جامعة غرب استراليا/ ولاية غرب استراليا/ استراليا. صاحب الحصول على شهادة الدكتوراه خبرة ميدانية ومختبرية وبرمجية متمثلة بإستحصال وقياس الخصائص الفيزيائية والهيدروليكية والثرموداينميكية لترب مختلفة. كما تم تحليل صور الإستشعار عن بعد (التحسس النائي) لتتبع حركة المياه من خلال العمل على لوغارتمات خاصة لتحليل البيانات. التعامل مع مثل هكذا لوغارتمات تمت خلال التمرس على إستخدام لغات البرمجة كلغة آر R ولغة بايثون $Python$ مما أعطى إمكانية تحديد صفات التربة من خلال الاستشعار عن بعد. فضلاً عن ذلك فقد تم التمكن من اللغة الانكليزية بصورة ممتازة لمواكبة التطور العلمي الموجود في بلد الدراسة إذ تم إلقاء محاضرات وإجراء مناظرات علمية كما سيتم تبيانه لاحقا.

شهادة الماجستير أُخذت من جامعة تكريت – كلية الزراعة – قسم التربة والموارد المائية. إذ تم العمل على نسب مختلفة من الجبس وأثر الترطيب والتجفيف المتعاقب على بعض الصفات الفيزيائية والميكانيكية. خلال فترة الماجستير تم قياس أغلب الخصائص الفيزيائية وبعض الخصائص الكيميائية للتربة الجبسية والتعرف عن عمق لأهم المشاكل التي تواجه الزراعة عند التعامل مع الترب الجبسية كخيار مستقبلي للزراعة. تم إستخدام برنامج الاكسل بكثافة لمعاجلة وتحليل جميع البيانات والحصول على النتائج الإحصائية.

البكالوريوس كان لها الدور الاساس في الدفع نحو إكمال الدراسات العليا للمؤلف. إذ تم الحصول على الشغف في العلوم الزراعية منذ الوهلة الأولى والدفع بأتجاه تبني أخر ما توصل اليه العلم من تقنيات وتطويرها محليا. تم غكمال دراسة البكالوريوس في جامعة تكريت – كلية الزراعة – الانتاج النبات العام.

ملخص الخبرة *Professional Summary*

- خبير في قياسات وتحاليل فيزياء التربة والخصائص الهيدرولوكية)حركة المياه في التربة(المختبرية والحقلية بالطرق المباشرة وأيضاً عن طريق التحسس النائي (صور الإستشعار عن بعد) بالطرق غير المباشرة.
- تحليل ودراسة صور التحسس النائي وكذلك صور الاشعة السينية للتتبع حركة المياه من خلال التعامل مع اللوغارتمات البرمجة ال R ولغة البرمجة ال $Python$.
- إستخدام أجهزة القياس الجيوفيزيائية لفحص خصائص التربة على اعماق مختلفة المتمثلة بالاجهزة الاتية:

$a-$ *Electronic resistivity tomography* (ERT)
$b-$ *Electromagnetic induction* (EMI)

الدورات العلمية *Courses*

- دورة تدريسية حول النظم الجغرافية.

GIS Programming University of Western Australia 01/2017 *to* 06/2017.

- دورة تدريسية حول كيفية اجراء معادلات بيئية.

Environmental Modelling University of Western Australia 07/2015 *to* 12/2015.

- لغة برمجة آر.

Advanced level of R Coding Courses including work with and modified several algorithms analysis codes.

- المشاركة في ورشة عمل حول شبكات الانهر والمجاري المائية في دولة كوريا الجنوبية.

Internarial Symposium on Dynamics of Structure and Functions of Complex Networks South Korea/Korea University $29-6-2015$ *to* $16-7-2015$.

- دورة مكثفة لدراسة اللغة الأنكليزية أكاديمياً.

Academic English Bridging Course :
Part 1 *and Part* 2 *Crawley, UWA* 07/2013 *to* 07/2014 *with High Distinction results.*

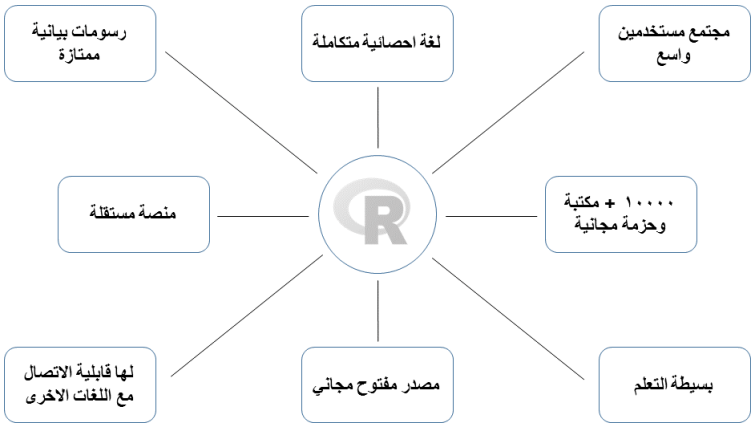
- دورة تدريبية حول ادارة المزارع النموذجية في الأردُن.

Using Technolog; Modern Techniques in management for Model Farms and Fields Organized by Japan International Cooperation Agency (JICA) under the third Country Training Program and implemented by (CARDNE) Conducted in Amman – Jordan (2012), *scoring a total of* 102 *training hours.*

Conferences المؤتمرات

- مؤتمر في دولة النمسا لعرض مشروع البحث الأول والثاني.

a –
European Geosciences Union General Assembly 2017 *Alsih, Abdulkareem. Gavan McGrath, Matthias Leopold. Experimental Investigation of 2D thermal signature and 3D XRay Computed Tomography in contrasting Wettable and Water Repellent Beads.*

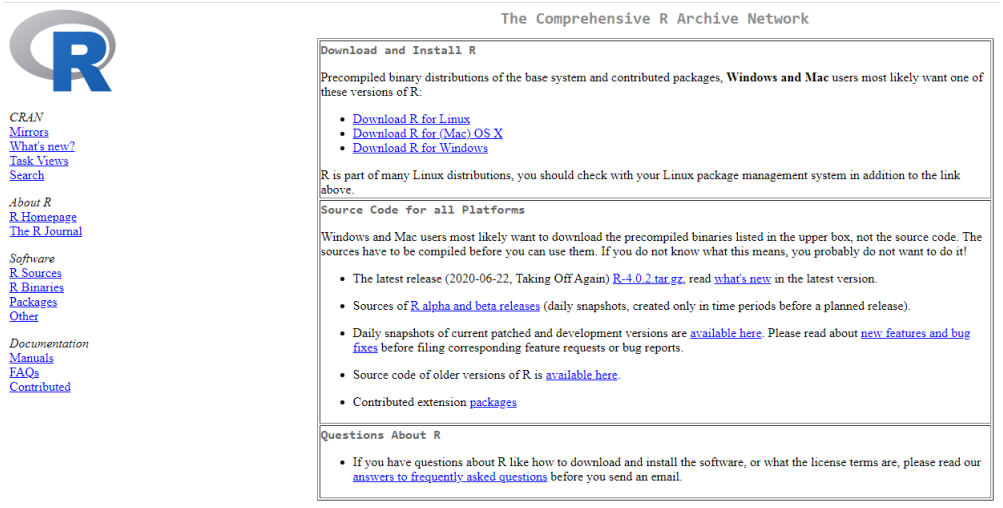


شكل ١.١: فوائد إستخدام لغة ال R

قبل الانطلاق والخوض في مفردات اللغة وجب علينا أولا الشرح قليلاً حول تنزيل وتنصيب هذه اللغة في الحاسوب, وهي طريقة ليست بذات الصعوبة مقارنة مع اللغات الأخرى كما سيأتي شرحه لاحقا. يقتصر شرح التحميل والتنصيب هنا على نظامين فقط هما نظام الويندوز $Windows$ ونظام الماك $macOS$.

١.١ تحميل ال آر *R download*

الخطوة الأولى للبدأ في إستخدام لغة ال R هي بتنزيلها إلى جهاز الحاسوب. البرنامج سهل الحصول عليه من الموقع الرسمي لأدارة لغة ال R والتي تسمى $The\ Comprehensive\ R\ Archive\ Network$ وإختصاره $CRAN$ على الرابط الآتي $https://cloud.r-project.org/$ كما موضح في الصورة أدناه.



شكل ١.٢: الصفحة الرسمية لتحميل ال R

يوجد في اعلى الصفحة للموقع ثلاث روابط يتم من خلالها تنصيب البرنامج (رابط للوندوز ورابط للماك ورابط لينكس). لمستخدمي الوندوز وهم الغالبية في بلدنا العزيز العراق يجب الضغط على الرابط المخصص للوندوز وستم تنصيب برنامج كتابة ال R أذ سيتم ملاحظة وجود ارقام تكتب هكذا $R4.x.x$ تُعبر عن النسخ المحدثه لهذا البرنامج. النسخه الاخيرة المحدثة عند كتابة هذا الكتاب هي R 4.0.2. حجم البرنامج لهذه النسخة الاخيره هو 84 $megabytes$ وتحمل على كل من 32 و 64 bit.

١.٢ مقارنة بين $32-bit$ و $64-bit$

الاختيار بين 64 bit و 32 bit ياتي عندما يكون هناك نظام يدعم 64 bit والتي أغلب الانظمة الحديثة تدعمها وكذلك حجم البيانات الواجب التعامل معها عند إستخدام لغة الR. بالنسبة إلى 64 bit بامكانها أن تعالج كمية بيانات كبيرة لذلك فهي تفضل أن وجدت في نظام الحاسوب لديك. من المهم جداً عند البدأ بأستخدام لغة الR إستخدام النسخة R 3.0.0 فما فوق لأنها تدعم التعامل مع دوال ومتغيرات ذات سعات حفظ عالية مقارنة مع النسخ القديمة. في السابق كان هناك بعض الحزم $packages$ تستخدم نظام 32 bit ولكن هذا أصبح قليل جداً في وقتنا الحاضر وشبه مندثر. لذلك تجدر الإشارة إلى أن السبب الوحيد لإختيار نظام 32 bit هو فقط عندما يكون نظام الحاسوب لديك لا يدعم نظام 64 bit.

١.٣ التنصيب $Installation$

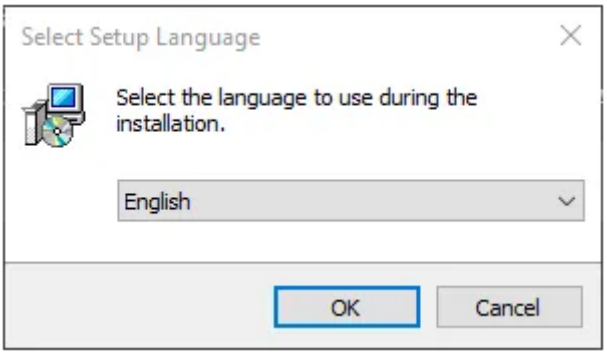
تنصيب ال R على نظام الوندوز ونظام الماك هو مشابه لتنصيب أي برنامج أخر. فالطريقة لا تخلو من اتباع الارشادات خطوة بخطوة من خلال الضغط على الزر $Next$ إذا أردت الاستمرار والزر $Back$ إذا اردت الرجوع خطوة الى الوراء والزر $Cancel$ إذ أردت الغاء العملية بكاملها.

١.٣.١ التنصيب على نظام الوندوز

بعد الانتهاء من تحميل ال R يجب النظر إلى المكان المخصص لتحميل البرامج على الحاسوب وعادة في نظام الوندوز تحمل جميع البرامج ضمن مجلد $Downloads$. بعد الضغط على ايقونة التحميل للبرنامج الموضحة بالشكل الآتي يبدأ البرنامج بالتنصيب.

شكل ١.٣: ايقونة التحميل

نافذة الحوار الأولى التي ستظهر هي لإختيار اللغة. اللغة المختارة هي اللغة الانكليزية إفتراضياً, أضغط على زر Ok ما لم تريد تغيير اللغة.



شكل ١.٤: إختيار اللغة

يدعم تنصيب ال R أكثر من 16 لغة حول العالم, في الغالب اللغات التي تدعمها ال R هي من أكثر لغات العالم إستخداما. من الممكن تغيير لغة التنصيب بعد تثبيت برنامج ال R من خلال تنفيذ الامر الاتي:

```
> Sys.setenv(LANG = "fr")
> 2 + x
Erreur : objet 'x' introuvable
> Sys.setenv(LANG = "en")
> 2 + x
Error: object 'x' not found
```

بمجرد تثبيت $BiocManager$ بنجاح, يمكنك تثبيت أي حزمة من $Bioconductor$ بإستخدام الدالة$BiocManager :: install()$, على سبيل المثال:

```
> BiocManager::install("ArrayTools")
```

Github Packages

يُعد $Github$ المستودع الأكثر شيوعًا للمشاريع مفتوحة المصدر. إنه شائع لأنه يأتي من المساحة غير المحدودة للمصدر المفتوح, والتكامل مع git, وبرنامج التحكم في الإصدار, وسهولة مشاركته والتعاون مع الآخرين.

$Installing\ from\ GitHub$ بينما لا يزال $CRAN$ هو المستودع الأكثر شيوعًا لحزم R, ستجد الكثير من الحزم لا تتوفر إلا من خلال المستودع $GitHub$. علاوة على ذلك, إذا كنت ترغب في تجربة أحدث إصدارات التطوير من الحزم الشائعة مثل $ggplot2$ و $tidyr$ [Wickham et al.2020] , فسيتعين عليك تثبيتها من خلال $GitHub$.

قبل الشروع بتنزيلأي حزمة من خلال $GitHub$, يتوجب عليك اولا تنزيل حزمة ال $remotes$ من مستودع ال $CRAN$. [Hester et al.2020]

```
> install.packages("remotes")
```

يمكنك الآن تثبيت أي حزمة من $GitHub$ عن طريق توفير "اسم المستخدم / المستودع" كوسيطة $remotes :: install_github()$. على سبيل المثال, لتثبيت أحدث إصدار تطوير من $ggplot2$ من $GitHub$, قم بتشغيل هذا الأمر الآتي:

```
> remotes::install_github("tidyverse/ggplot2")
```

$R - Forge\ Packages$

يُعد $R - Forge$ مكان آخر مثير للاهتمام للبحث عن الحزم. يحتوي موقع $R - Forge$ على مشاريع قيد التنفيذ, ويوفر أدوات للمطورين للتعاون. قد تجد بعض الحزم المثيرة للاهتمام على هذا الموقع, ولكن يرجى التأكد من قراءة إخلاء المسؤولية والوثائق, لأن العديد من هذه الحزم قيد التشغيل. يمكن الوصول إلى الموقع من خلال الرابط الآتي $http://r-forge.r-project.org/$. $install\ from\ R - Forge$ عند الرغبة في تثبيت احد الحزم الموجودة في $R - forge$ على سبيل المثال حزمة $MPAgenomics$ [Grimonprez et al.2014] , فيجب تحديد عنوان URL الخاص بمشروع $R-Forge$ في وسيطة $repos$ الخاصة بدالة $install.packages$. يتم إستخدام دالة التبعيات (المتعلقات) $dependencies$ عندما لا تكون repos فارغة, لتحديد ما إذا كان يجب تثبيت حزم ثانوية أو متعلقة بالحزمة الاساس أو لا.

```
> install.packages("MPAgenomics", repos = "http://R-Forge.R- +  project.
   org",dependencies = TRUE)
```

Packages from Other Sources

هناك العديد من الحزم موجودة في مصادر أخرى تُعد اقل انتشاراً من المصادر السابقة الذكر. من المصادر (المواقع) التي يوجد بها الحزم ومن الممكن تحميلها أيضاً هي *Gitlab, SVN and Bitbucket* الى حاسوبك الشخصي. يمكن التحميل من خلال حزمة ال *remotes*() والتي تحتوي على العديد من الدوال القادرة على تحميل الحزم من مختلف المواقع كما في المثال الآتي:

```
 > grep(
+   pattern = "^install_",
+   x = getNamespaceExports("remotes"),
+   value = TRUE
+ )
 [1] "install_gitlab"    "install_url"       "install_github"
 [4] "install_svn"       "install_cran"      "install_version"
 [7] "install_deps"      "install_bioc"      "install_bitbucket"
[10] "install_git"       "install_local"     "install_remote"
[13] "install_dev"
```

في المثال أعلاه تم سرد الدوال التي من الممكن أن تُستخدم لتحميل (تنزيل) الحزم من المواقع المختلفة.

Packages from ZIP Source

ربما تكون قد قمت بتنزيل حزمة بصيغة *zip* أو *tar.gz*, وربما قد تم ارسالها لك من قبل صديق. لتثبيت الحزمة من ملف مضغوط محلي, تحتاج فقط إلى استدعاء دالة *install.packages* مع المدخلات مساوية ل $repos = NULL$ واكتب $type = "source"$. مع ملاحظة أن مسار الملف لا يجب أن يحتوي على مسافات.

```
> install.packages("file_path\\package_file_name.extension",
+                  repos = NULL, type = "source")
```

قبل هذا الاجراء من الضروري تغيير مسار ملف العمل *Folder Path* من الحالي إلى مكان تحميل الحزم (الموجود ضمن الملفات داخل مجلد ال *R*). يتم تغيير المسار بخطوتين الأولى معرفة المسار الحالي وذلك عن طريق الدالة *getwd*() والثانية بتغيير المسار وذلك بكتابة إسم المسار داخل الدالة *setwd*() كما في المثال الآتي:

```
> getwd()
[1] "\\userhome/students9/21273229/My Documents"
> setwd("\\Libraries\\Documents\\My Documents\\R\\win-library\\4.1")
```

```
> help(solve)
## An alternative is
> ?(solve)
```

في كثير من تنزيلات ال R فقرة المساعدة تأتي بصيغة ال $HTML$ عند تنفيذ الامر الآتي:

```
> help.start()
```

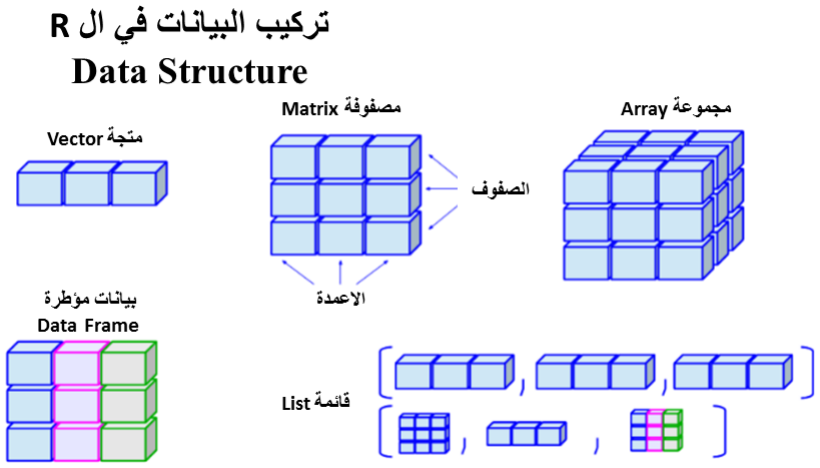
والتي ستطلق متصفح ويب يسمح بتصفح صفحات المساعدة بالارتباطات التشعبية. يُعد ارتباط "محرك البحث والكلمات الرئيسة" في الصفحة التي تم تحميلها بواسطة $help.start()$ مفيدًا بشكل خاص لأنه يحتوي على قائمة مفاهيم عالية المستوى تبحث من خلال الوظائف المتاحة. هذه الطريقة مهمة جداً لفهم طبيعة عمل الدوال واختصار للوقت والجهد بالبحث عن طبيعة تطبيق مثل هكذا دوال.

يمكن أدراج أغلب الأوامر المفيدة عند إستخدام ال R في القائمة الاتية مع شرح مبسط لكل امر. تم ترك التعليقات في القائمة باللغة الانكليزية للتاكيد على القارئ أن من المهم جداً التعامل مع المصطلحات الانكليزية وتداولها قدر الممكن لانها تمثل الجزء الاكبر في فهم لغة البرمجة في R واللغات الاخرى ايضا.

```
#Command      #   Purpose
>help()       #   Obtain documentation for a given R command
>example()    #  View some examples on the use of a command
>c(),
>scan()       #  Enter data manually to a vector in R
>seq()        #  Make arithmetic progression vector
>rep()        #  Make vector of repeated values
>data()       #  Load (often into a data.frame) built-in >dataset
>View()       #  View dataset in a spreadsheet-type format
>str()        #  Display internal structure of an R object
>read.csv(),
>read.table() #  Load into a data.frame an existing data file
>library(),
>require()    #  Make available an R add-on package
>dim()        #  See dimensions (# of rows/cols) of data.frame
>length()     #  Give length of a vector
>ls()         #  Lists memory contents
>rm()         #  Removes an item from memory
>names()      #  Lists names of variables in a data.frame
>hist()       #  Command for producing a histogram
>histogram()  #  Lattice command for producing a histogram
>stem()       #  Make a stem plot
>table()      #  List all values of a variable with frequencies
>xtabs()      #  Cross-tabulation tables using formulas
>mosaicplot() #  Make a mosaic plot
>cut()        #  Groups values of a variable into larger bins
>mean(),
>median()     #  Identify ""center of distribution
>by()         #  apply function to a column split by factors
>summary()    #  Display 5-number summary and mean
>var(),
>sd()         #  Find variance, sd of values in vector
>sum()        #  Add up all values in a vector
>quantile()   #  Find the position of a quantile in a dataset
>barplot()    #  Produces a bar graph
>barchart()   #  Lattice command for producing bar graphs
>oxplot()     #  Produces a boxplot
>bwplot()     #  Lattice command for producing boxplots
>plot()       #  Produces a scatterplot
>xyplot()     #  Lattice command for producing a scatterplot
>lm()         #  Determine the least-squares regression line
>anova()      #  Analysis of variance (can use on results of lm())
>predict()    #  Obtain predicted values from linear model
>nls()        #  estimate parameters of a nonlinear model
>residuals()  #  gives (observed - predicted) for a model fit to data
>sample()     #  take a sample from a vector of data
```

- البيانات المؤطرة *Data Frame*.
- القوائم *List*.
- العوامل *Factors*.

هذه التراكيب تكون إما متجانسة في التركيب *homogeneous* أو مختلفة (متنوعة) في التركيب *heterogeneous*. البيانات المتجانسة في التركيب هي المتجة والمصفوف على سبيل المثال (لأحتوائها على نوع واحد من البيانات), والمختلفة في التركيب هي القوائم (لأحتوائها على أكثر من تركيب أو نوع من البيانات). يُعد المتجة من ابسط التراكيب البيانية في ال *R*. من الممكن التعامل مع المتجة بسهولة كما لا يتطلب معرفة وجهد في انشائه. كما ويجب الإشارة إلى أن في بعض الأحيان التعامل مع البيانات يتطلب امور أكثر تعقيدا من استخدام المتجهات فقط. فهناك تراكيب بيانات تسهل على المتعاملين مع الأرقام العديد من العمليات الحسابية وكذلك ترتيبها في أطر معينة تسهم في سهولة الوصول إلى القيم واستخراجها واجراء العمليات الحسابية عليها. من أهم التراكيب البيانية المتقدمة هنا هي المصفوفة, المجموعة, البيانات المؤطرة والقوائم.



شكل ٤.٣: تركيب البيانات في ال *R*

٤.٢.١ المتجهات *Vectors*

في علم الحاسوب وخصوصا في عالم اللغات مفهوم المتجهات يختلف قليلاً عما تم أخذه في دروس الرياضيات. في علم الرياضيات يعبر عن المتجة بكمية وأتجاه معين أما في لغة البرمجة فهو عبارة فقط عن مجموعة ارقام ضمن متغير معين.

لا يكون للمتجه في البرمجة عادةً أي استخدام هندسي, مما يعني أنك غير مهتم حقًا بأشياء مثل الحجم أو الاتجاه. المتجه في البرمجة هو بالأحرى مجرد قائمة مرتبة من العناصر. المهم هنا هو أن العناصر لا يلزم أن تكون أرقامًا – يمكن التعامل مع أي شيء بواسطة اللغة المعنية! وبالتالي,

$("I","Love","Iraq")$ هي أيضًا متجه في البرمجة ، بالرغم من أنها (من الواضح) ليس لها تفسير متجه بالمعنى الرياضي. تحصر قيم المتجة بالعلامة $c()$ فكل شيء داخل هذه العلامة يُعد متجة. كما ويمكن أن ينشأ المتجة من أكثر من طريقة كما مبين في الشكل ادناه.

الدالة Function	مثال Example	النتيجة Result
c(a, b, ...)	c(1, 3, 6 ...)	1, 3, 6
a:b	1:4	1, 2, 3, 4
seq(from, to, by, length.out)	seq(from = 0, to = 8, by = 2)	0, 2, 4, 6, 8
rep(x, times, each, length.out)	rep(c(1, 2), times = 2, each = 2)	1, 1, 2, 2, 1, 1, 2, 2

شكل ٤.٤: طرق إنشاء المتجة المختلفة

يعمل ال R على هياكل البيانات المختلفة (انوع البيانات المختلفة). إذ من الممكن أن يتواجد متجهات رقمية, متجهات نصية ومتجهات منطقية.

المتجهات الرقمية *Numeric Vactors*

من ابسط هياكل البيانات هي ما تسمى المتجهات الرقمية *Numeric Vectore*, وهي عبارة عن كيان معين تحتوي قيم معينة. فعلى سبيل المثال يمكن أعداد متجة بأسم x, يتكون من خمسة أرقام على النحو الآتي:

```
> x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
```

العلامة التي تأتي بعد x يجب أن تُكتب ويمكن الإستعاضة عنها أيضاً بالرمز $=$. فلا يمكن البدء بأي عملية بسيطة أو معقدة دون استخدام احدهما.

العلامة التي خصصت القيم للمتجة x يمكن أن تكتب بشكل مختلف كما موضح أدناه:

```
> c(10.4, 5.6, 3.1, 6.4, 21.7)-> x
> assign("x", c(10.4, 5.6, 3.1, 6.4, 21.7))
```

قيمة ال x لا تتغير اذا ما اجريت عليها عمليات حسابية مباشرة فعلى سبيل المثال, اذا تم ضرب x بالرقم 2 ستظهر لدينا ارقام مضروبة بالرقم 2 دون تغيير بمحتوى x الاصلي من القيم.

```
 > 2*x
[1] 20.8 11.2  6.2 12.8 43.4
 > x
[1] 10.4  5.6  3.1  6.4 21.7
```

كما ويمكن ايجاد متجه آخر يحتوي 11 رقم من خلال احتوائة على المتجه x كما في المثال الآتي:

```
 > y<- c(x, 0, x)
 > y
 [1] 10.4  5.6  3.1  6.4 21.7  0.0 10.4  5.6  3.1  6.4 21.7
```

توجد طرق أخرى لأنشاء المتجة فعلى سبيل المثال توجد دوال مثل $rep()$ و $seq()$ تعمل على تكوين متجة بناء على المدخلات. فالدالة $rep()$ تسمح بتكرار المتجة أو الرقم عدد من المرات أو يمكن تحديد عدد التكرار من خلال مدخل $lenght.out$.

الدالة *rep(x, times, each, length.out)*	
المدخل (Argument)	التعريف (Definition)
x	القيمة او المتجة المراد تكراره
times	عدد مرات التكرار
each	عدد تكرار كل قيمة
length.out	طول المتسلسلة للمتجة

شكل ٤.٥: دالة ال $rep()$ ومدخلاتها

كما في المثال الآتي:

```
rep(x = 3, times = 10)
##  [1] 3 3 3 3 3 3 3 3 3 3
rep(x = c(1, 2), each = 3)
## [1] 1 1 1 2 2 2
rep(x = 1:3, length.out = 10)
##  [1] 1 2 3 1 2 3 1 2 3 1
```

أما دالة $seq()$ فتعمل على انتاج متسلسلة رقمية (متجة) تبدأ بقيمة وتنتهي بقيمة على أن يتم تقسيمها بواسطة المدخل by كما في الشكل الآتي:

الدالة *seq(from, to, by, length.out)*	
المدخل (Argument)	التعريف (Definition)
from	البداية (رقم البداية)
to	النهاية (رقم النهاية)
by	خطوات القفز (التخطي)
length.out	طول المتسلسلة للمتجة

شكل ٤.٦: دالة ال $seq()$ ومدخلاتها

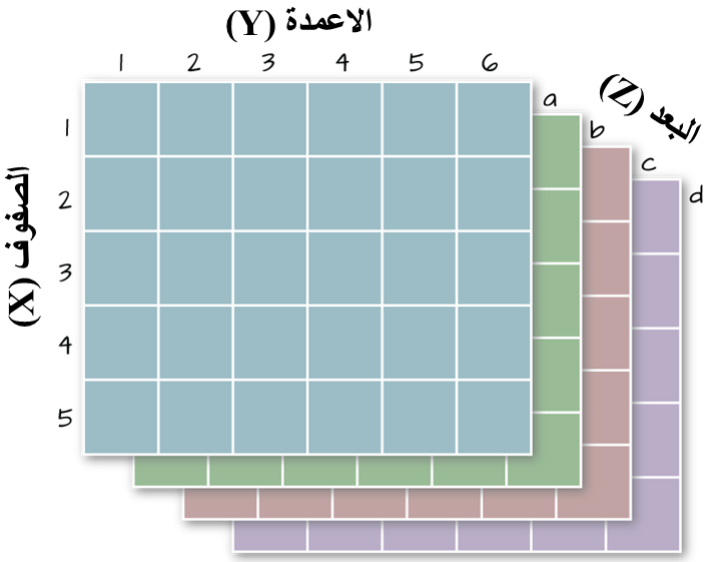
يمكن تمثيل ما تم شرحهُ بالمثال الآتي:

```
> # Create the numbers from 1 to 10 in steps of 1
> seq(from = 1, to = 10, by = 1)
 [1]  1  2  3  4  5  6  7  8  9 10
>
> # Integers from 0 to 100 in steps of 10
> seq(from = 0, to = 100, by = 10)
 [1]   0  10  20  30  40  50  60  70  80  90 100
```

```
  return(z)
}
> outer(x4, y4, Fun)
     [,1] [,2] [,3] [,4] [,5]
[1,]   16   25   36   49   64
[2,]   25   36   49   64   81
[3,]   36   49   64   81  100
[4,]   49   64   81  100  121
[5,]   64   81  100  121  144
```

٤.٢.٦ المجموعة *Array*

المجموعة هي عبارة عن مصفوفة متعددة الابعاد, فعلى سبيل المثال تتكون المصفوفة من بعدين بينما تتكون المجموعة من ثلاث ابعاد أو اكثر. لكي نقرب الصورة أكثر للقارئ الكريم نأخذ مثال الصور الفوتغرافية فهي تتكون من ثلاث مصفوفات (مجموعة) أن صح التعبير متكونة من الالوان (الأحمر, الأزرق والأخضر). فكل لون هو مصفوفة في حد ذاته متكون من قيم تختلف بأختلاف شدة الضوء. يعبر عن الالوان الثلاثة بالمجموعة كونها تعطي تركيب بياني من ثلاث أبعاد لتنتج عنه ما يعرف بالصورة الفوتغرافية.



شكل ٤.١٠: المجموعة بشكل مبسط

أنشاء المجموعة *Create An Array*

من الممكن إنشاء مجموعة بعدة طرق من ابسطها واكثرها شيوعا هي من خلال الامر $array()$ ويمكن الاستدلال على المدخلات لهذه الدالة من خلال الامر $array()$?إذ سيظهر لنا الامر الآتي:

$$ArrNmae <- array(data, dim = (row_size, column_size, matrices, dimnames$$

- $data$ البيانات التي ستحول إلى مجموعة.
- row_size عدد الصفوف لكل مصفوفة في المجموعة.
- $column_size$ عدد الأعمدة لكل مصفوفة في المجنوعة.
- $matrices$ عدد المصفوفات في المجموعة.
- $dimnames$ أسماء الأعمدة والصفوف في المصفوفات المكونة للمجموعة.

المثال الآتي يوضح ابسط الطرق لأنشاء المجموعة من خلال إنشاء مجموعة متكونة من مصفوفتين:

```
> # Create
>
> A <- array(1: 24, dim = c(3, 4, 2))
> print(A)
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

, , 2

     [,1] [,2] [,3] [,4]
[1,]   13   16   19   22
[2,]   14   17   20   23
[3,]   15   18   21   24

> vect1 <- c(10, 20, 30, 40)
> vect2 <- c(50, 60, 70, 80, 90, 100)
> B <- array(c(vect1, vect2), dim = c(3, 3, 2))
> print(B)
, , 1

     [,1] [,2] [,3]
[1,]   10   40   70
[2,]   20   50   80
[3,]   30   60   90

, , 2

     [,1] [,2] [,3]
[1,]  100   30   60
[2,]   10   40   70
```

```
> a <- 32
> b <- 6
> add <- a + b
> sub = a - b
> multi = a * b
> division = a / b
> Integer_Division = a %/% b
> exponent = a ^ b
> modulus = a %% b
> print(paste("Addition of two numbers", a, "and", b, "is : ", add))
[1] "Addition of two numbers 32 and 6 is :  38"
> print(paste("Subtracting number", a, "from", b, "is : ", sub))
[1] "Subtracting number 32 from 6 is :  26"
> print(paste("Multiplication of two numbers",a, "and", b, "is : ", multi))
[1] "Multiplication of two numbers 32 and 6 is :  192"
> print(paste("Division of two numbers", a, "and", b, "is : ", division))
[1] "Division of two numbers 32 and 6 is :  5.33333333333333"
> print(paste("Integer Division of two numbers", a, "and", b, "is : ",
    Integer_Division))
[1] "Integer Division of two numbers 32 and 6 is :  5"
> print(paste("Exponent of two numbers", a, "and", b, "is : ", exponent))
[1] "Exponent of two numbers 32 and 6 is :  1073741824"
> print(paste("Modulus of two numbers", a, "and", b, "is : ", modulus))
[1] "Modulus of two numbers 32 and 6 is :  2"
```

٥.٢ المشغلات الترابطية *Relational Operators*

يتم استخدام المشغلات الترابطية (المقارنة) في برمجة R غالبًا إما في حالة الشروط $if\ condition$ أو الحلقات $Loops$. تُستخدم المشغلات الترابطية بشكل شائع للتحقق من العلاقة بين متغيرين.

شكل ٥.٤: المشغلات الترابطية

يوضح الشكل الآتي المشغلات الترابطية التي تدعمها لغة R. علما أن المشغلات تعمل على كل عنصر من عناصر المتجه.

المشغل Operator	الوصف Description	مثال Example
>	للتحقق مما إذا كان كل عنصر من عناصر المتجه الأول أكبر من العنصر المقابل للمتجه الثاني.	v <- c(2,5.5,6,9) t <- c(8,2.5,14,9) print(v>t) [1] FALSE TRUE FALSE FALSE
<	للتحقق مما إذا كان كل عنصر من عناصر المتجه الأول أقل من العنصر المقابل للمتجه الثاني.	v <- c(2,5.5,6,9) t <- c(8,2.5,14,9) print(v < t) [1] TRUE FALSE TRUE FALSE
==	للتحقق مما إذا كان كل عنصر من عناصر المتجه الأول مساوي الى العنصر المقابل للمتجه الثاني.	v <- c(2,5.5,6,9) t <- c(8,2.5,14,9) print(v == t) [1] FALSE FALSE FALSE TRUE
<=	للتحقق مما إذا كان كل عنصر من عناصر المتجه الأول أقل من او مساوي العنصر المقابل للمتجه الثاني.	v <- c(2,5.5,6,9) t <- c(8,2.5,14,9) print(v<=t) [1] TRUE FALSE TRUE TRUE
>=	للتحقق مما إذا كان كل عنصر من عناصر المتجه الأول أكبر من او يساوي العنصر المقابل للمتجه الثاني.	v <- c(2,5.5,6,9) t <- c(8,2.5,14,9) print(v>=t) [1] FALSE TRUE FALSE TRUE
!=	للتحقق مما إذا كان كل عنصر من عناصر المتجه الأول لا يساوي العنصر المقابل للمتجه الثاني.	v <- c(2,5.5,6,9) t <- c(8,2.5,14,9) print(v!=t) [1] TRUE TRUE TRUE FALSE

شكل ٥.٥: شرح المشغلات الترابطية مع الامثلة

يساعد المثال الآتي على فهم طبيعة عمل المشغلات الترابطية (المقارنة) في لغة البرمجة R عمليًا. في المثال, يستخدم متغيرين هما a و b والقيم الخاصة بهما هي 15 و 12. سنستخدم هذين المتغيرين لإجراء عمليات ارتباط مختلفة موجودة في *R Programming*.

```
> # Example for Comparison Operators in R Programming
> a <- 15
> b <- 12
> print(paste("Output of 15 > 12 is : ", a > b))
[1] "Output of 15 > 12 is :  TRUE"
> print(paste("Output of 15 < 12 is : ", a < b))
[1] "Output of 15 < 12 is :  FALSE"
> print(paste("Output of 15 >= 12 is : ", a >= b))
[1] "Output of 15 >= 12 is :  TRUE"
> print(paste("Output of 15 <= 12 is : ", a <= b))
[1] "Output of 15 <= 12 is :  FALSE"
> print(paste("Output of 15 Equal to 12 is : ", a == b))
[1] "Output of 15 Equal to 12 is :  FALSE"
> print(paste("Output of 15 Not Equal to 12 is : ", a != b))
[1] "Output of 15 Not Equal to 12 is :  TRUE"
```

٥.٣ المشغلات المنطقية *Logical Operators*

تُستخدم المشغلات المنطقية في برمجة R للجمع بين شرطين أو أكثر, وتنفيذ العمليات المنطقية باستخدام & بمعنى (مع), إبمعنى (أو) و ! بمعنى (ليس).

شكل ٥.٦: المشغلات المنطقية

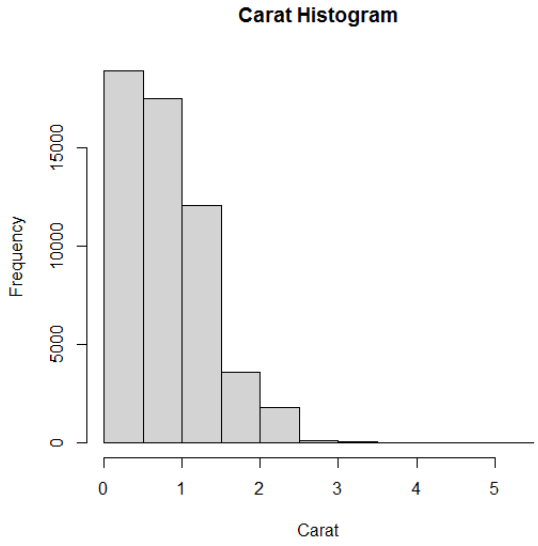
يوضح الشكل الآتي المشغلات المنطقية التي تدعمها لغة R. علما أن المشغلات تعمل على كل عنصر من عناصر المتجه.

المشغل Operator	الوصف Description	مثال Example
&	يجمع كل عنصر من المتجه الأول مع العنصر المقابل للمتجه الثاني ويعطي ناتجًا TRUE إذا كان كلا العنصرين TRUE.	v <- c(3,1,TRUE,2+3i) t <- c(4,1,FALSE,2+3i) print(v&t) [1] TRUE TRUE FALSE TRUE
|	يجمع كل عنصر من المتجه الأول مع العنصر المقابل للمتجه الثاني ويعطي ناتجًا TRUE إذا كان أحد العناصر هو TRUE	v <- c(3,0,TRUE,2+2i) t <- c(4,0,FALSE,2+3i) print(v|t) [1] TRUE FALSE TRUE TRUE
!	يأخذ كل عنصر من المتجه ويعطي القيمة المنطقية المعاكسة.	v <- c(3,0,TRUE,2+2i) print(!v) [1] FALSE TRUE FALSE FALSE
&&	يأخذ العنصر الأول من كلا المتجهين ويعطي TRUE فقط إذا كان كلاهما TRUE.	v <- c(3,0,TRUE,2+2i) t <- c(1,3,TRUE,2+3i) print(v&&t) [1] TRUE
||	يأخذ العنصر الأول من كلا المتجهين ويعطي TRUE فقط إذا كان احدهما TRUE.	v <- c(0,0,TRUE,2+2i) t <- c(0,3,TRUE,2+3i) print(v||t) [1] FALSE

شكل ٥.٧: شرح المشغلات المنطقية مع الامثلة

للتعمق أكثر بالمشغلات المنطقية يجب معرفة تفاصيل عملها مع امثلة تلامس فائدتها الاساسية. تقوم المشغلات الترابطية بمقارنة متغيرين فقط, أما اذا اردنا المقارنة بين أكثر من متغيرين فما علينا سوى استخدام المشغلات المنطقية. فعلى سبيل المثال, المشغل && يعمل على تحديد القيم ما بين متغيرين لأرجاع ال $TRUE$ أن كانت حقيقية. أما المشغل |فيعمل على اعطاء خيارين اتباع الأول أو الثاني كما في المثال الآتي:

```
> # Logical Operators in R example
> age <- 16
> if (!(age > 18)) {
+   print("You are Too Young")
+ } else if(age > 18 && age <= 35) {
+   print("Young Guy")
+ } else if(age == 36 || age <= 60) {
+   print("You are Middle Age Person")
+ } else {
+   print("You are too Old")
+ }
[1] "You are Too Young"
```



شكل ٧.١: الرسم البياني ل قيم القيراط في الالماس

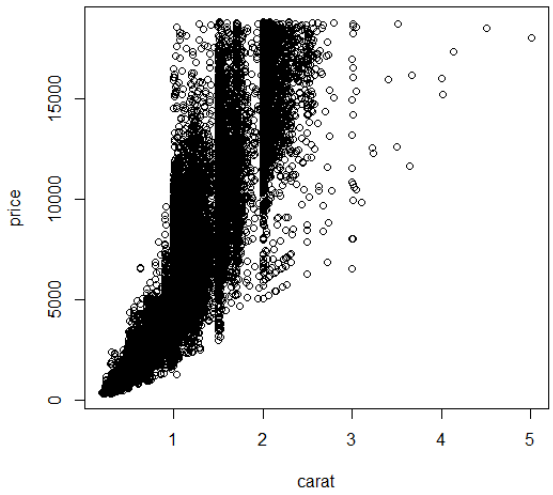
يوضح الرسم توزيع احجام القيراط للالماس. من المهم ملاحظة أن الاسم للمحور السيني تم كتابتهُ عن طريق المدخل $xlab$. هذا الرسم البياني في ابسط الصور وممن المكن الخوض أكثر ي امور تفصيلية ومتقدمة ولها القليل من التعقيد في الرسوم البيانية الموجودة في حزمة ال $ggplot2$.

٧.١.٢ الرسوم المبعثرة الاساسي *Base Scatterplot*

من الجيد والمهم جداً أن ترى العلاقة بين متغيرين في رسم توضيحي. إذ تمثل النقطة الواحدة على الرسم المبعثر العلاقة بين المحور السيني والصادي وتكون ممثلة لهما جميعا. من هنا يمكننا أن نقوم برسم العلاقة بين السعر والقيراط من خلال استعمال $formula\ notation$

```
plot(price ~ carat, data = diamonds)
```

الكود المكتوب انفا ينتج عنه الرسم البياني الآتي:



شكل ٧.٢: الرسم المبعثر للأماس مبينا العلاقة بين السعر والقيراط

علامة الفاصلة بين السعر والقيراط في الكود السابق تعمل على إظهار العلاقة بين المدخلين (السعر والقيراط) إذ أن السعر يوضع في المحور الصادي والقيراط في المحور السيني. المعادلة التي على أساسها وجدت العلاقة بين المحور السيني والصادي والمتمثلة بالعلامة سيتم شرحها في الفصول القادمة.

من الممكن ايجاد العلاقة بين القيم في المحور السيني والقيم في المحور الصادي بدون اللجوء إلى المعادلة $formula$ وذلك باستخدام العلاقة المباشرة كما يأتي:

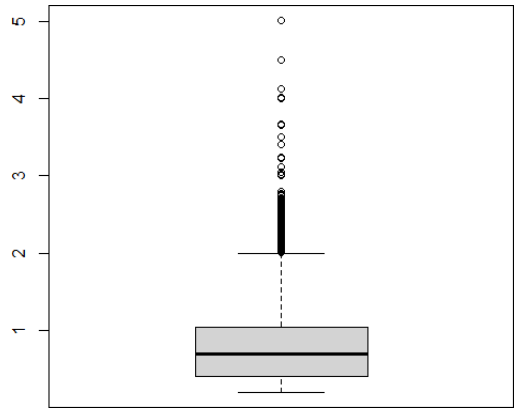
```
> plot(diamonds$carat, diamonds$price)
```

يُعد الرسم المبعثر من أهم الرسوم التي تُستخدم في العلاقات أو البيانات الإحصائية وسيتم الاسهاب به أكثر عندما نصل إلى حزمة ال $ggplot2$.

٧.١.٣ الرسم الصندوقي *Boxplot*

بالرغم من أن الرسم الصندوقي من اوائل الرسومات التي تُدرس لطلاب الإحصاء والحساب, إلا إنها لا تزال تحت المناقشة والتداول في مجتمع اهل الاختصاص. إذ ينقسم اهل الاختصاص إلى صفين ما بين مؤيد لها ولأهميتها ومابين معارض لها [Mahto2020]. بالرغم من ذلك فأن لغة ال R توفر دالة للتعامل مع الرسم الصندوقي كما يأتي:

```
boxplot(diamonds$carat)
```



شكل ٧.٣: الرسم الصندوقي لقيراط الالماس

الفكرة الأساسية من وراء الرسم الصندوقي تتمثل بأستحصال مدى انتشار القيم, القيم الدنى والعليا, تماثل البيانات وانحرافها, القيم المتطرفة والربيعيات $Quartiles$.
كما اوضح سابقا فأن الرسم الصندوقي يوجد أيضاً في حزمة ال $ggplot2$ بصورة أكثر تقدما.

٧.١.٤ الرسم الدائري $Pie\ Chart$

يُعد الرسم البياني واحد من أهم انواع الرسوم البيانية التي يستطيع ال R تنفيذها بسهولة. يعرض الرسم الدائري البيانات على شكل شرائط تكون في مجملها دائرة بنسبة مئة في المئة.
في ال R يتم إنشاء الرسم الدائري من خلال دالة ال $pie()$ والتي تأخذ قيم موجبة كمدخلات. باقي المدخلات تتحكم في اللون وتسمية المحاور وغيرها.

$$pie(x,\ labels,\ radius,\ main,\ col,\ clockwise)$$

- x عبارة عن متجة رقمي يستعمل لأعطاء قيم كل شريط.
- $labels$ تُستخدم لأعطاء وصف لكل شريط من الاشرطة في الرسم الدائري.
- $radius$ تعمل على تحديد نصف قطر الرسم الدائري.
- $main$ تشير إلى إنشاء عنوان رئيسي للرسم الدائري.
- col تشير إلى الالوان (لوحة الاوان) للرسم الدائري.
- $clockwise$ قيمة منطقية تشير إلى ما إذا كانت الشرائح مرسومة في اتجاه عقارب الساعة أو عكس اتجاه عقارب الساعة.

يتم إنشاء الرسم الدائري البسيط من خلال الاتي:

- دالة ال $qplot()$: تعمل على اخراج رسم سريع بسيط وسهل للمستخدم.
- دالة ال $ggplot()$: دالة أكثر مرونة وقوة من $qplot()$ وتُسهم في بناء الرسم خطوة بخطوة.

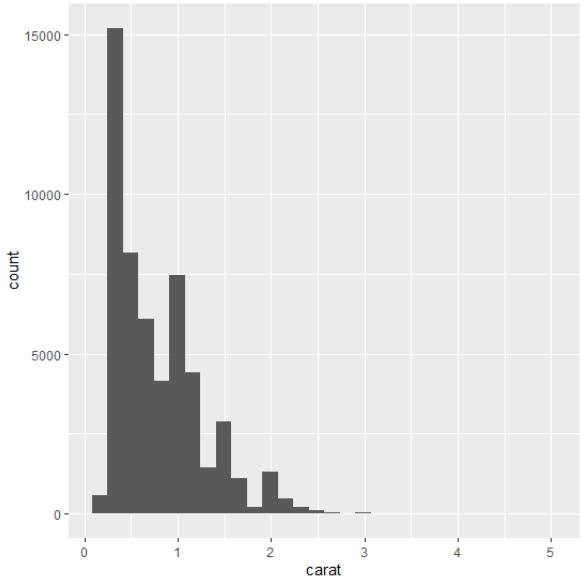
٧.٢.١ الرسوم البيانية والكثافات $Histogram\ and\ Densities$

بنفس صيغة العمل للرسوم البيانية في قسم الرسوم الاساسية, تقوم حزمة $ggplot2$ برسم توزيع القيراط للماس. يمكن عمل ذلك بخطوتين الأولى تحديد البيانات بواسطة الدالة $ggplot()$ والثانية بتحديد نوع الرسم بواسطة الدالة $geom_histogram()$. بما أن الرسم البياني ذو بعد واحد فقط لذلك تم استخدام محور واحد فقط وهو الصادي $x - axis$.

```
> require(ggplot2)
> ggplot(data = diamonds) + geom_histogram(aes(x = carat))
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

ظهور رسالة تحذيرية ليس بالضرورة أن يكون ما قمت به خطأ وانما هناك نقص ما في البيانات أو لأعلامك بوجود شي معين اهمل أو اضيف.
فعلى سبيل المثال في هذه الحالة ظهرت لنا رسالة تقول أن $bins = 30$ وهي عدد الأعمدة المستخدمة يجب تحديدها في الدالة أو الكود المكتوب. لتجنب الامر نكتب الاتي:

```
> ggplot(data = diamonds) + geom_histogram(aes(x = carat), bins = 30)
```

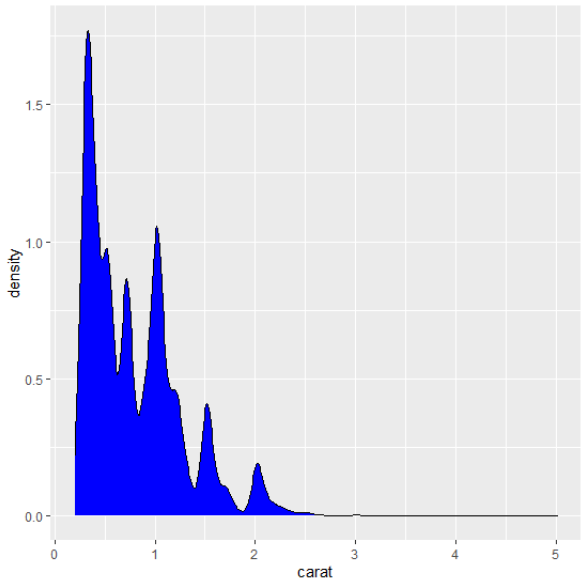


شكل ٧.١٠: الرسم البياني لقيراط الالماس بأستخدام حزمة ال $ggplot2$

ستظهر نفس النتائج عندما يتم رسم الكثافة, وذلك عند تغيير الدالة $geom_histogram$ إلى $geom_density$.
كما ويمكن تحديد اللون بأستخدام المدخل $fill$. من المهم ملاحظة أن المدخل $fill$ وضع خارج الدالة

aes وذلك لأننا نرغب بتغيير لون كل الرسم.

```
> ggplot(data = diamonds) + geom_density(aes(x = carat), fill = "blue")
```



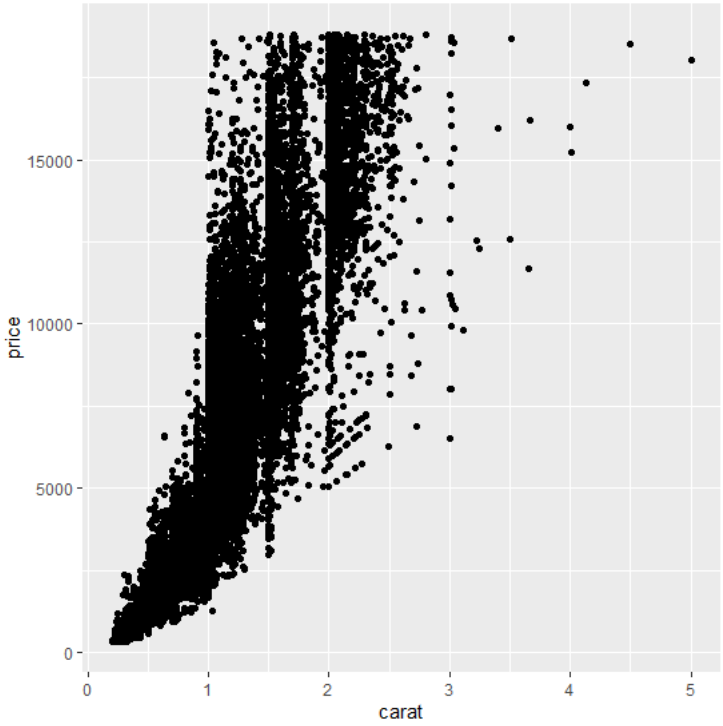
شكل ٧.١١: الرسم البياني لقيراط الالماس بأستخدام حزمة ال *ggplot2*

بينما الرسوم البيانية توضح البيانات ضمن نطاق المستطيلات, تقوم الرسوم الكثافية بتوضيح البينات احتماليا بين اثنين أو أكثر من المديات. الفرق بين الاثنين قليل ولكنه مهم جدا. الرسوم البيانية حدية القياس جداً برسم البيانات اما الرسوم الكثافية فهي متسلسلة القياس.

٧.٢.٢ الرسومات المبعثرة *Scotterplot*

نحن هنا لا نعمل على إظهار كيفية عمل الرسومات بواسطة حزمة ال *ggplot2* ولكن نظهر هنا المقدرة الكبيرة لهذه الحزمة على التعامل مع مختلف الاحتياجات للرسومات المختلفة. سنقوم بأعادة الرسم البياني السابق ولاك في هذه الحالة سنستخدم الرسم المبعثر. الرسم المبعثر هو رسم ذوبعدين متكون من العلاقة بين المحور السيني (القيراط) $x-axis$ والمحور الصادي (السعر) $y-axis$. لذلك في دالة ال $aes()$ سيتم إدراج كل من $(x,\ y)$.

```
> ggplot(data = diamonds, aes(x = carat, y = price))+ geom_point()
```



شكل ٧.١٢: رسم مبعثر بسيط

في الفقرات اللاحقة سيتم استخدام الكود أعلاه بكثرة وبصورة متكررة لذلك من الافضل أن يتم تخصيصهُ إلى متغير للسهولة كما يأتي:

```
# save a commonly used code to a variable
> g <- ggplot(data = diamonds, aes(x = carat, y = price))
```

الان يمكننا أن نتقدم أكثر في هذا الرسم من خلال اضافة الطبقات اللازمة من خلال استخدام علامة الزائد + بعد المتغير g. أستخدام دالة $grom\_point()$ تعمل على ادخال الكثير من المدخلات للتلاعب بالبيانات الموجودة في الرسم مثل اللون والحجم وغيرها.

```
> g + geom_point(aes(color = color))
```

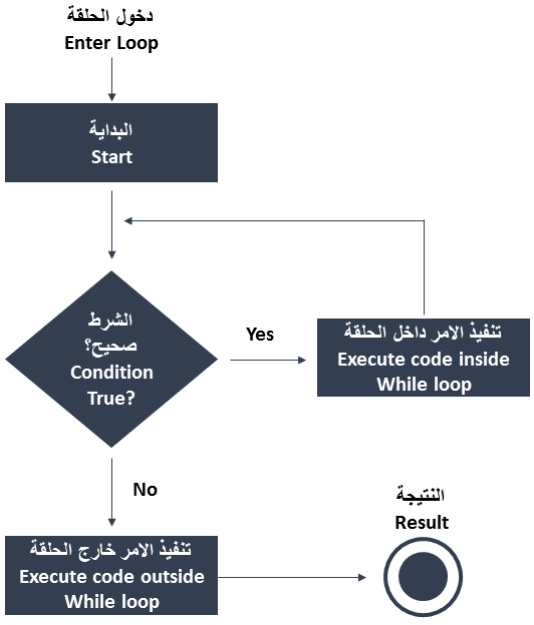
```
+         print(paste("Found:", first_name[i], last_name[i]))
+         break
+     }
+ }
[1] "Found: Ali Mohamed"
```

في المثال أعلاه هناك ثلاث نتائج يمكن تفسيرها كالاتي:

- النتيجة الأولى: عبارة عن جملة شرطية داخل حلقة تشترط وجود إسم علي كأسم الشخص الأول ومن ثم طباعة باقي المعلومات لكل علي في القائمة
- النتيجة الثانية: احتوت هذه على فقرة $next$ والتي تعني القفز إلى القيمة الآتية للفحص بسرعة.
- النتيجة الثالثة: عبارة عن استعمال ال $break$ لأيقاف الحلقة عند وجود اول إسم لعلي (أو القيمة المحددة في الجملة الشرطية)

٩.٢ *while loop*

بالرغم من إنها تستعمل اقل بكثير من ال $for\ loop$ في ال R, تُعد ال $while\ loop$ احد اسهل الحلقات استخداما. ببساطة تقوم بتنفيذ الامر داخل الاقواس مادام نتيجة الاختبار هي صحيحة $TRUE$.



شكل ٩.٣: مخطط يوضح طبيعة عمل حلقة ال $while\ loop$

الصيغة التنفيذية ل $while\ loop$ في ال R هي كاالاتي:

$$while\ (< condition >)\ < action >$$

- $< condition >$ شرط منطقي, يجب أن يكون اما $TRUE$ أو $FALSE$.
- $< action >$ أمر التنفيذ طالما أن الشرط المنطقي مساويا ل $TRUE$.
-

في المثال الآتي, نقوم بطباعة قيمة ال x وتكرارها حتى نصل إلى قيمة 5 حسب الشرط الموجود بين القوسين. يُعد المثال الآتي من الامثلة البسيطة جداً لأستخدام $while\ loop$ ويمكن وضعه هنا للتوضيح.

```
> x <- 1
> while (x <= 5)
+ {
+     print(x)
+     x = x + 1
+ }
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

لكي يتم هضم الموضوع أكثر نقوم هنا بطرح احد الامثلة التطبيقية ل $while\ loop$. فعلى سبيل المثال نود طباعة جميع ارقام x والتي تبدا من $1, 2, 3...\infty$ على شرط أن تكون قيمة x^2 اقل من 20.

```
> # Start with 0
> x <- 0
> # Loop until condition is FALSE
> while (x^2 < 20) {
+   print(x)      # Print x
+   x <- x + 1    # Increase x by 1
+ }
[1] 0
[1] 1
[1] 2
[1] 3
[1] 4
```

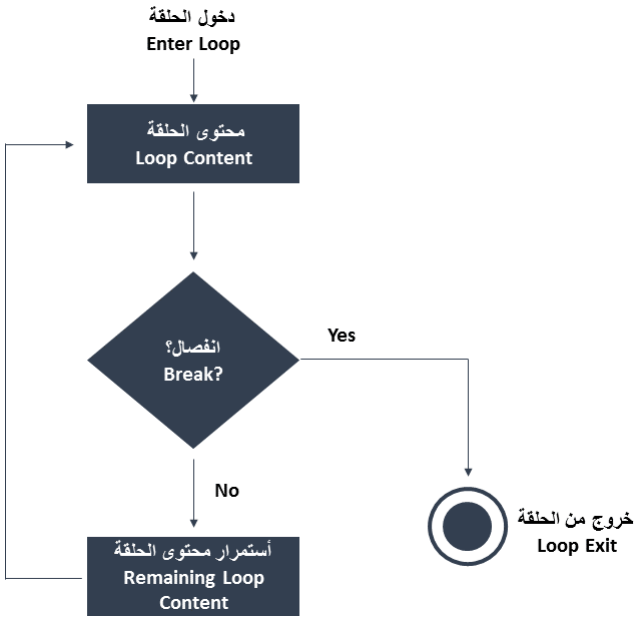
يفسر أمر التنفيذ في المثال أعلاه على النحو الآتي:

- العنصر x مساويا ل 0 بالتالي 0^2 اقل من 20 مما يحقق الشرط ليكون $TRUE$ لذلك تستمر قيمة x بالزيادة
- العنصر x مساويا ل 1 بالتالي 1^2 اقل من 20 مما يحقق الشرط ليكون $TRUE$ لذلك تسمتر قيمة x بالزيادة

- العنصر x مساويا ل 2 بالتالي 2^2 اقل من 20 مما يحقق الشرط ليكون $TRUE$ لذلك تسمتر قيمة x بالزيادة
- العنصر x مساويا ل 3 بالتالي 3^2 اقل من 20 مما يحقق الشرط ليكون $TRUE$ لذلك تسمتر قيمة x بالزيادة
- العنصر x مساويا ل 4 بالتالي 4^2 اقل من 20 مما يحقق الشرط ليكون $TRUE$ لذلك تسمتر قيمة x بالزيادة
- العنصر x مساويا ل 5 بالتالي 5^2 أكثر من 20 مما لايحقق الشرط ليكون $FALSE$ لذلك يتم ايقاف الحلقة

٩.٣ *repeat loop*

تقوم حلقة ال *repeat* بالتكرار خلال أمر معين عدد من المرات. كما وتُعد حلقة ال *repeat* من الحلقات التي لا تتوقف إلا إذا تم إعطاء الامر داخلياً بالتوقف عن طريق ال *break* كما هو موضح في المخطط الآتي:



شكل ٩.٤: مخطط يوضح طبيعة عمل حلقة ال *repeat loop*

في المثال الآتي ستم استخدام حلقة ال *repeat* والتي لن تتوقف مالم يتم استخدام كلمة *break* والتي تعني الانفصال والخروج من الحلقة.

من أجل الخوض أكثر في كيفية التعامل مع دالة ال *merge()* سنقوم بعمل بيانات مصطنعة. الأول هو *dum_1* وهو عبارة مجموعة بيانات تحتوي على رقم الشخص *id*, الاسم *name* والراتب الشهري *monthly salary* لبعض الموظفين. اما الثاني *dum_2* عبارة عن مجموعة بيانات تحتوي على رقم الشخص *id*, الاسم *name*, العمر *age* والمنصب *Postion* لبعض الموظفين كما مبين ادناه:

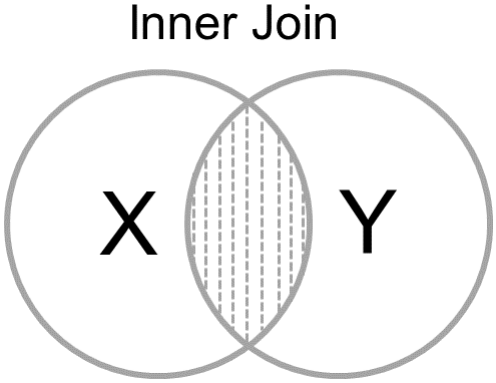
```
> set.seed(61)
>
> employee_id <- 1:10
> employee_name <- c("Ali", "Reem", "Khalid", "Moaz", "Waiel",
+                    "Mostafa", "Fatiema", "Manal", "Noor", "Qaseem")
> employee_salary <- round(rnorm(10, mean = 1500, sd = 200))
> employee_age <- round(rnorm(10, mean = 50, sd = 8))
> employee_position <- c("CTO", "CFO", "Administrative", rep("Technician",
   7))
> dum_1 <- data.frame(id = employee_id[1:8], name = employee_name[1:8],
+                     month_salary = employee_salary[1:8])
> dum_2 <- data.frame(id = employee_id[-5], name = employee_name[-5],
+                     age = employee_age[-5], position = employee_position
   [-5])
>
> dum_1
  id    name month_salary
1  1     Ali         1424
2  2    Reem         1425
3  3  Khalid         1156
4  4    Moaz         1570
5  5   Waiel         1223
6  6 Mostafa         1462
7  7 Fatiema         1641
8  8   Manal         1603
> dum_2
  id    name age        position
1  1     Ali  40             CTO
2  2    Reem  38             CFO
3  3  Khalid  54 Administrative
4  4    Moaz  66      Technician
5  6 Mostafa  38      Technician
6  7 Fatiema  53      Technician
7  8   Manal  56      Technician
8  9    Noor  55      Technician
9 10  Qaseem  43      Technician
```

عند التعامل مع البيانات الحقيقية يجب ملاحظة أن الاسم يمكن أن يتكرر ولكن رقم الشخص *id* هو رقم لايمكن تكرارة ابدا فلكل شخص رقم معين. كما يجب أن يلاحظ أن *Waiel* مفقود في البيانات الثانية *dum_2* كما ينطبق ذلك على كل من *Qaseem* و *Noor* إذ لا يوجد لهم ذكر في البيانات

الأولى *dum_1*.
الان من الممكن الخوض في دمج البيانات بصور مختلفة حسب رغبة المبرمج من خلال العناوين الآتية:

Inner join

يُعد هذا النوع من الدمج من أكثر الانواع استخداماً وانتشاراً إذ يعمل على دمج البيانات المتشابهة في البيانات المؤطرة التي يرغب في دمجها.أي بمعنى يتم اهمال البيانات الفردية التي تذكر في احد البيانات وليس لها ذكر في البيانات الأخرى مثل *Qaseem* و *Noor* التي توجد لهما بيانات في الثنانية ولا يوجد لهم أثر في الأولى كما في المثال اعلاه.



شكل ١٢.١: اندماج المتشابهات

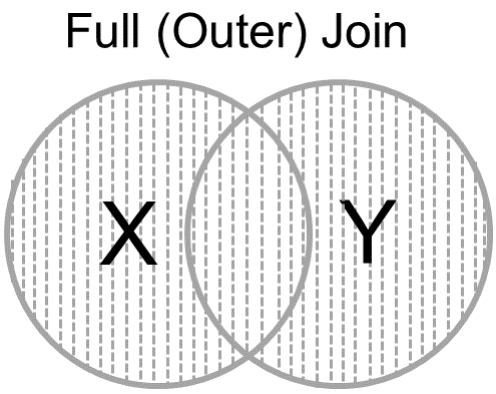
الشكل أعلاه يوضح البيانات التي سيتم دمجها والتي تكون في الغالب مشتركة بين مجموعتين من البيانات. من أجل فعل ذلك, ما علينا سوى وضع البيانت ضمن دالة ال *merge()* وكاالاتي:

```
> merge(dum_1, dum_2)
  id     name month_salary age       position
1  1      Ali          1424  40            CTO
2  2     Reem          1425  38            CFO
3  3   Khalid          1156  54 Administrative
4  4     Moaz          1570  66     Technician
5  6  Mostafa          1462  38     Technician
6  7  Fatiema          1641  53     Technician
7  8    Manal          1603  56     Technician
> merge(dum_1, dum_2,by = c("id", "name")) # Equivalent
  id     name month_salary age       position
1  1      Ali          1424  40            CTO
2  2     Reem          1425  38            CFO
3  3   Khalid          1156  54 Administrative
4  4     Moaz          1570  66     Technician
5  6  Mostafa          1462  38     Technician
6  7  Fatiema          1641  53     Technician
7  8    Manal          1603  56     Technician
```

كما اشرنا سابقا, الأسماء غير المشتركة لن يتم تضمينها في البيانات المخرجة.

Full (outer) join

الاندماج الكلي *Full join* يعرف أيضاً بالاندماج الخارجي *Outer join* عبارة عن اتحاد جالبيانات المؤطرة (اثنان أو اكثر) في جدول واحد يشمل الكل كما موضح في الشكل ادناه.



شكل ١٢.٢: الاندماج الكلي

من أجل دمج اثنان من البيانات المؤطرة بطريقة الدمج الكلي ما عليك سوى تتشخيص المدخل *all* وجعله مساويا ل *TRUE*.

```
> merge(dum_1, dum_2, all = TRUE)
   id    name month_salary age       position
1   1     Ali          1424  40            CTO
2   2    Reem          1425  38            CFO
3   3  Khalid          1156  54 Administrative
4   4    Moaz          1570  66     Technician
5   5   Waiel          1223  NA           <NA> # <-- NA values
6   6 Mostafa          1462  38     Technician
7   7 Fatiema          1641  53     Technician
8   8   Manal          1603  56     Technician
9   9    Noor            NA  55     Technician # <-- NA values
10 10  Qaseem            NA  43     Technician # <-- NA values
```

وجود قيم ال *NAs* دلالة على عدم التوافق بين المجموعتين من البيانات, أي أن الاسم موجود في احدى البيانات ولكن لايوجد في الاخر.

Right (outer) join

اندماج الجهة اليمنى يعني إدراج جميع بيانات السطور (والقيم المرادفة لها) من البيانات الأولى مع بيانات السطور المتطابقة (المشابهة) لها من البيانات الثانية كما موضحة في الشكل الآتي:

الاسم $John$ فلا نستطيع استخدام الدالة str_sub لأيجادة بل سنعمل مع الدالة str_detect كما يأتي:

```
> JohnPre <- str_detect(string = Presidents$PRESIDENT, pattern = "john")
> Presidents[JohnPre,c("YEAR", "PRESIDENT", "Start", "Stop")]
     YEAR PRESIDENT Start Stop
NA   <NA>      <NA>    NA   NA
```

يلاحظ هنا أن الدالة لم تفلح بايجادأي إسم وهذا منافي للحقيقة, اذن أين تكمن المشكلة؟. كما هو معلوم ويجب الانتباه اليه دائما انه ال R من اللغات الحساسة للحروف الكبيرة والصغيرة وبما أن المدخل $pattern()$ في دالة $str_detect()$ أخذ الاسم $john$ بالحرف الصغير وهو موجود بالحرف الكبير بالجدول اذن فأكيد لن نحصل على شيء. لتصحيح ذلك هو بتغيير الحرف الصغير بالحرف الكبير $john - John$.

```
> JohnPre <- str_detect(string = Presidents$PRESIDENT, pattern = "John")
> Presidents[JohnPre,c("YEAR", "PRESIDENT", "Start", "Stop")]
          YEAR          PRESIDENT Start Stop
X     1797-1801         John Adams  1797 1801
X.8   1825-1829 John Quincy Adams  1825 1829
X.13  1841-1845         John Tyler  1841 1845
X.22  1865-1869     Andrew Johnson  1865 1869
X.51  1961-1963    John F. Kennedy  1961 1963
X.52  1963-1965 Lyndon B. Johnson  1963 1965
X.53  1965-1969 Lyndon B. Johnson  1965 1969
```

لأظهار الكثير من الأمور الهامة للتعبيرات الاعتيادية سنستخدم جدول آخر أيضاً من الويكيبيديا, الحروب في الولايات المتحدة الامريكية. للسهولة تم تحميل وخزن البيانات داخل الحاسوب بمسمى $warTimes.rdata$ ومن ثم تم تحميليها على ال R لكي يتم التعامل معها.

```
> setwd("C:\\Users\\21273229\\Desktop\\Book(R)")
> load("warTimes.rdata")
> head(warTimes, n = 10)
 [1] "September 1, 1774 ACAEA September 3, 1783"
 [2] "September 1, 1774 ACAEA March 17, 1776"
 [3] "1775ACAEA1783"
 [4] "June 1775 ACAEA October 1776"
 [5] "July 1776 ACAEA March 1777"
 [6] "June 14, 1777 ACAEA October 17, 1777"
 [7] "1777ACAEA1778"
 [8] "1775ACAEA1782"
 [9] "1776ACAEA1794"
[10] "1778ACAEA1782"
```

تحتوي البيانات على تاريخ بداية الحرب وتاريخ النهاية. في بعض الأحيان تحتوي فقط على السنة وفي احيان أخرى تحتوي على الشهر والسنة وأيضاً الايام. وهناك مفارقات قد تكون لسنة واحدة فقط. بسبب

هذا كان من المناسب وضعها في جدول وفهرستها جيدا.
نحتاج هنا لعمل عمود يحتوي على سنين بداية الحرب. لعمل ذلك نحتاج أيضاً إلى شيء مشترك بين جميع البينات, بدورنا نشكر تشفير (ترميز)الويكيبيديا لجعل ال $ACAEA$ صفة مشتركة بين جميع البيانات تقريبا. كما ويلاحظ وجود الشارحة $(-)$ بموقعين الأول تعمل على شكل فاصل بين الكلمات والآخر تعمل على شكل شارحة وصل اعتيادية كما في المثال الآتي:

```
> warTimes[str_detect(string = warTimes, pattern = "-")]
[1] "6 June 1944 ACAEA mid-July 1944" "25 August-17 December 1944"
```

من هنا نعلم اننا اذا اردنا فصل المتسلسلات النصية يجب علينا ايجاد عامل مشترك وهما اما الشارحة $(-)$ أو الرمز $ACAEA$. في دالة ال $str_split()$ المدخل $pattern$ من الممكن أن يأخذ تعبير اعتيادي يحتوي على كل من $(-)$ و $ACAEA$. لتجنب بعض الأمور مثل تواجد الشارحة كحد فاصل بين كلميتن $mid-July$ يتم تحديد المدخل n برقم 2 لينتج عندنا في الغالب قطعتين بدل القطعة الواحدة. كما ويتم تحديد الاولوية للفصل من خلال وضع التعبير بين قوسين مثل $(ACAEA)$.

```
> Times <- str_split(string = warTimes, pattern = "(ACAEA)|-", n = 2)
> head(Times)
[[1]]
[1] "September 1, 1774 " " September 3, 1783"

[[2]]
[1] "September 1, 1774 " " March 17, 1776"

[[3]]
[1] "1775" "1783"

[[4]]
[1] "June 1775 "    " October 1776"

[[5]]
[1] "July 1776 "  " March 1777"

[[6]]
[1] "June 14, 1777 "    " October 17, 1777"
```

يبدو أن العمل تم بصورة جيدة, إلا أن التأكد من المخرجات هي من شيم المبرمجين. نحدد هنا بدالة ال $which()$أي من التسلسلات النصية تحتوي على الشارحة كما في المثال الآتي:

```
> which(str_detect(string = warTimes, pattern = "-"))
[1] 147 150
```

تم اعلامنا من خلال دالة ال $wich()$أي الأرقام التسلسلية تحتوي على متسلسلة نصية يكون بها شارحة وقد اعطت النتائج تسلسل 147 و 150. الان يتم مقارنتها بالمخرجات من المثال السابق لنرى كيفية عمل الدالة على المتسلسلات النصية.

كما أن كتابة ال $[0-9]$ غير مريح وعملي في بعض الأحيان إذ يمكن أيضاً الاستعانة عنه بالرمز d .\\

```
> head(str_extract(string = StartWar, pattern = "\\d{4}"), n = 20)
 [1] "1774" "1774" "1775" "1775" "1776" "1777" "1777" "1775" "1776" "1778"
[11] "1775" "1779" NA     "1785" "1798" "1801" NA     "1812" "1812" "1813"
```

كما ويمكن أن نبحث عن قيم عدة تكون من 1 إلى 3 ارقام فيها.

```
> str_extract(string = StartWar, pattern = "\\d{1,3}")
  [1] "1"   "1"   "177" "177" "177" "14"  "177" "177" "177" "177" "177" "
      177"
 [13] NA    "178" "179" "180" NA    "18"  "181" "181" "181" "181" "181" "
     181"
 [25] "181" "181" "181" "181" "181" "181" "22"  "181" "181" "5"   "182" "
     182"
 [37] "182" NA    "6"   "183" "23"  "183" "19"  "11"  "25"  "184" "184" "
     184"
 [49] "184" "184" "185" "184" "28"  "185" "13"  "4"   "185" "185" "185" "
     185"
 [61] "185" "185" "6"   "185" "6"   "186" "12"  "186" "186" "186" "186" "
     186"
 [73] "17"  "31"  "186" "20"  "186" "186" "186" "186" "186" "17"  "1"   "6"
 [85] "12"  "27"  "187" "187" "187" "187" "187" "187" NA    "30"  "188" "
     189"
 [97] "22"  "189" "21"  "189" "25"  "189" "189" "189" "189" "189" "189" "2"
[109] "189" "28"  "191" "21"  "28"  "191" "191" "191" "191" "191" "191" "
    191"
[121] "191" "191" "191" "7"   "194" "194" NA    NA    "3"   "7"   "194" "
    194"
[133] NA    "20"  NA    "1"   "16"  "194" "8"   "194" "17"  "9"   "194" "3"
[145] "22"  "22"  "6"   "6"   "15"  "25"  "25"  "16"  "8"   "6"   "194" "
    195"
[157] "195" "195" "195" "197" "28"  "25"  "15"  "24"  "19"  "198" "15"  "
    198"
[169] "4"   "20"  "2"   "199" "199" "199" "19"  "20"  "24"  "7"   "7"   "7"
[181] "15"  "7"   "6"   "20"  "16"  "14"  "200" "19"
```

كما ويمكن استخلاص القيم في بداية السطر عن طريق اشارة $\wedge$ والقيم في نهاية السطر عن طريق اشارة $.

```
> # extract 4 digits at the begining of hte text
> head(str_extract(string = StartWar, pattern = "^\\d{1,4}"), n = 30)
 [1] NA     NA     "1775" NA     NA     NA     "1777" "1775" "1776" "1778"
[11] "1775" "1779" NA     "1785" "1798" "1801" NA     NA     "1812" "1813"
[21] "1812" "1812" "1813" "1813" "1813" "1814" "1813" "1814" "1813" "1815"
> # extract 4 digits at the end pf the text
```