Benqué, David. ‘Kernels Idiotiques’.
dans MàJ: Design, Environnements Techniques & Pratiques Exploratoires,
sous la direction de Gwenaëlle Bertrand, Maxime Favard, et David-Olivier Lartigaud. St. Etienne, France: Cité du Design, 2021.

David Benqué

Kernels idiotiques

«L’IA et le machine-learning permettent de laisser libre cours à votre créativité.» (Adobe, 2020)

Selon Adobe, la firme au monopole de l’industrie des logiciels de création, le design a commencé son «moment IA» (*AI moment*, traduction personnelle). Ce moment est décrit par les économistes Ajay Agrawal, Joshua Gans et Avi Goldfarb (2018) comme le tournant révolutionnaire que tout secteur d’activité économique aurait pris récemment, ou serait sur le point de prendre, grâce à l’intelligence artificielle (IA). Selon ces économistes, ce moment résulte de la conjonction entre une accumulation massive de données numériques, une application généralisée des techniques statistiques et algorithmiques de machine-learning (ML) et surtout la baisse du prix des opérations informatiques. Les promesses de l’intelligence artificielle dans ce discours, reprises en partie par Adobe, se résument à des gains de productivité et donc à des retours sur investissement. Mais la redéfinition de tout problème comme un problème de prédiction algorithmique (Agrawal *et al.*, 2018) a des conséquences culturelles, politiques et sociales profondes. Les ramifications du «moment IA» vont jusqu’à remettre en question la définition même du savoir, réduit à la donnée à travers des formes numériques de positivisme (Jurgenson, 2014) et de néoplatonisme (McQuillan, 2018).

Dans le design graphique et d’interaction, le tournant algorithmique prend différentes formes. L’augmentation du processus créatif promu par Adobe et son programme Sensei repose sur le fait de décharger sur une IA les tâches les plus fastidieuses — comme la recherche d’image ou le détourage — pour laisser l’humain donner libre cours à sa créativité. Également, de nouvelles fonctionnalités impossibles à réaliser par des humains, telles que la reconfiguration complète de paysages photographiés, ouvrent de nouvelles possibilités.

Le design est aussi chargé d’implémenter les «moment[s] IA» des autres disciplines ou secteurs d’activité, en particulier en donnant forme à des produits numériques. Le champ du «design anticipatif» (Shapiro, 2015) propose de déléguer aux algorithmes le maximum de décisions, les utilisateurs étant saturés par une avalanche de choix. Le design se trouve ici dans une position inconfortable puisqu’il doit défaire son propre rôle historique de hiérarchisation de l’information pour produire, à la place, des interfaces les plus «lisses» possible. Cette situation est poussée à l’extrême dans la conception des assistants vocaux tels que Siri (Apple) ou Cortana (Microsoft) qui mettent en narration l’imaginaire d’une intelligence artificielle personnifiée.

Enfin, l’industrie de l’intelligence artificielle est ravie de faire appel au design et à l’art pour faire des algorithmes un matériau de création. Des programmes d’exploration créative de ces «nouvelles technologies» (par ex. Google Arts & Culture et Google AI, 2019) permettent d’appliquer une couche cosmétique et culturelle apte à amplifier l’imaginaire d’une collaboration humain / machine, faisant une fois encore passer la machine pour autonome, sinon consciente. Dans la mesure où le designer se trouve tour à tour client, complice ou promoteur de l’intelligence artificielle, le «moment IA» caractérise une subordination toujours plus importante de la discipline (comme tant d’autres) aux *dispositifs* algorithmiques (Masure, 2014).

Néanmoins, dans cet article, je propose une approche diamétralement opposée au Sensei d’Adobe et autres imaginaires d’une intelligence artificielle. Dans cette position, le design se saisit des dispositifs algorithmiques comme des objets dont les composants et opérations doivent être manipulés, sondés, contestés, détournés, en bref critiqués. Cette approche emprunte en partie au travail critique déjà avancé de longue date par les chercheurs en sciences humaines, de la théorie des médias à l’histoire des sciences. Il s’agit, pour le design, de contribuer à ces débats en y apportant une sensibilité et une *pratique* alors que tant d’analyses critiques théoriques opèrent en hauteur, sans contact avec les matériaux qui forment les dispositifs (code, données, appareils, etc.). Suivant le *Manifeste médiarchéologiste* (Bardini *et al.*, 2016), je propose que le designer doit «[opérer] des descentes souterraines dans les couches matérielles des médias» et se faire archéologue des dispositifs algorithmiques passés et présents. Je vais commencer par expliquer le projet *almanac.computer*, situé entre design graphique, archéologie des médias et techniques algorithmiques. Ce projet revisite le format de l’almanach comme un prototype des systèmes algorithmiques actuels, je m’en sers comme d’un point d’entrée dans l’histoire de la prédiction algorithmique et comme appui pour la remettre en question aujourd’hui. L’intérêt est ici d’exposer l’un des outils avec lesquels j’ai fabriqué mon almanach. Il s’agit d’un format de programmation informatique particulier, le Jupyter Notebook, qui encourage une approche exploratoire et narrative du code. J’inscris cet outil et l’utilisation que j’en fait avec *almanac.computer* dans une position théorique plus large et en faveur d’une pratique du design critique des systèmes algorithmiques. Je définis cette posture avec l’aide de la figure de l’idiot, conceptualisée par Gilles Deleuze et Félix Guattari (1991) puis par Isabelle Stengers (2007), qui résiste aux interprétations ambiantes et produit un trouble créatif. Je propose une pratique du design plongée dans les opérations algorithmiques, pour mieux désamorcer les récits hégémoniques faits de/avec l’IA. En particulier, le noyau (*kernel*) du Jupyter Notebook semble présenter un terrain propice aux explorations absurdes et critiques. L’idiot refusant toute marche à suivre, il ne peut donc s’agir ici que de potentiels à discerner et à ouvrir, plutôt que de recettes à appliquer.

L’imaginaire cosmique de la prédiction algorithmique : *almanac.computer*

Le projet *almanac.computer*[1] fait partie de mes recherches de doctorat sur les diagrammes et la prédiction algorithmique (Benqué, 2020). Sa première itération s’est déroulée lors d’un workshop mené pour les étudiant·e·s de l’Esadse en 2016[2]. Il s’agissait de revisiter l’almanach, une publication oubliée et insolite, pour explorer la présentation de systèmes de croyance sous forme de visualisations de données. J’ai ensuite développé le projet dans les années suivantes, pour en faire une publication automatisée en ligne qui produit chaque jour un nouveau numéro.

Les almanachs sont des guides pratiques pour naviguer dans un futur incertain, en général l’année à venir. On y trouve calendriers, cartes, prévisions, tables de conversion, miscellanées, folklores et autres informations utiles au quotidien. Bien que des artefacts similaires, tablettes, registres et cartes, existent depuis des millénaires dans de nombreuses cultures, les almanachs imprimés sont en circulation depuis le XVII^e siècle. Ils sont encore publiés aujourd’hui dans des domaines tels que l’agriculture, la navigation nautique ou la finance.

Selon la journaliste Adrienne LaFrance (2015), *The Old Farmer’s Almanac* peut être vu comme un prototype de l’internet. Cette publication annuelle remplissait les mêmes fonctions au XVIII^e siècle qu’un smartphone aujourd’hui avec par exemple la consultation de prévisions météorologiques. Selon Adrienne LaFrance, au-delà des fonctionnalités, l’almanach occupe un «espace culturel» similaire aux réseaux d’aujourd’hui. Les almanachs sont le produit d’un imaginaire où le cosmos influence

directement les événements sur Terre (Benqué, 2018). Les modèles de prévisions météorologiques du *Old Farmer's Almanac* sont basés sur l'activité des taches solaires. Plus généralement, on retrouve dans les pages des almanachs l'origine des méthodes statistiques inventées en premier lieu pour la pratique de l'astronomie. Elles y côtoient les cycles lunaires ainsi que des croyances divinatoires telles que l'astrologie. Depuis le premier numéro du *Old Farmer's Almanac,* en 1792, certaines parties de cet imaginaire cosmique des données et de la prédiction ont été amplifiées et légitimées, elles sont aujourd'hui appelées « analyse de données » ou « intelligence artificielle ». D'autres, telles que l'astrologie, ont été écartées comme des formes illégitimes de spéculation, en contradiction avec une rationalité scientifique.

Almanac.computer revisite l'almanach pour sonder l'imaginaire cosmique qui entoure les données et les algorithmes. Le projet utilise les outils de la *data science* contemporaine au service de rationalités divinatoires — des formes d'astrologie algorithmique. Les données astronomiques, tirées de diverses éphémérides, servent de base pour des prédictions allant des marchés financiers à l'infrastructure énergétique, en passant par les tirages du loto et les tâches quotidiennes. *Almanac.computer* cherche à montrer l'histoire de la prédiction sous un nouvel angle, mettant en évidence la médiation et l'interprétation qui s'opèrent dans des dispositifs algorithmiques souvent présentés comme neutres et objectifs. Dans *almanac.computer,* toute prédiction est une mise en relation entre forces cosmiques et événements sur Terre : cela interroge la légitimité accordée à certaines formes de spéculation et refusée à d'autres.

En pratique, *almanac.computer* consiste en une série de programmes qui produisent chacun une section de la publication en ligne. Ces scripts mêlent l'outillage algorithmique contemporain [3] et des rationalités divinatoires comme l'astrologie. Ainsi, la section « Cosmic Commodity Charts » présente des prédictions des prix des marchandises (blé, maïs, bétail, jus d'orange) sur les marchés spéculatifs en fonction des positions des planètes du système solaire. Une autre section, « Crisis Proximity Index », propose une astrologie basée sur un événement originel : la crise financière de 2008. La position des planètes est interprétée selon leur proximité avec leur position le 9 août 2007, date du gel de trois fonds d'investissements par BNP Paribas, signe avant-coureur de la crise.

Le développement de ces programmes s'est fait par tâtonnements, en expérimentant avec des données disparates telles que les cours boursiers et les positions astronomiques, jusqu'à réussir à produire des relations statistiques entre elles. Les outils de programmation ont un niveau d'abstraction élevé, c'est-à-dire que des notions mathématiques sont emballées dans des fonctions simples à utiliser. Il est donc possible d'appliquer des opérations complexes, notamment en algèbre linéaire, sans comprendre complètement leur fonctionnement interne. Depuis ma position, dans une démarche créative et critique, ce procédé d'expérimentation a donc comporté une part d'attention à ce que les programmes produisaient au fur et à mesure des commandes, suivant tour à tour des intuitions ou répondant aux surprises. Le mode de programmation utilisé, Jupyter Notebook, a grandement facilité ce processus. Il est devenu une partie intégrante du projet, manifestant lui aussi certaines caractéristiques de ce que pourrait être un almanach algorithmique contemporain.

Le notebook, un mode de programmation exploratoire

Si l'on suit, en tant que designer, l'injonction de se « mettre à jour » sur les techniques algorithmiques afin de ne pas rater le « moment IA », on rencontre assez vite le Jupyter Notebook. Ce format est, en effet, très utilisé à des fins pédagogiques. Il permet d'inclure des explications et illustrations en parallèle de programmes et de détailler leurs opérations. Il est facilement publiable sur internet, ce qui en fait un format privilégié pour de nombreux cours (par ex. Downey, 2020), livres (par ex. Davidson-Pilon, 2015), conférences ou autres tutoriels (par ex. Kogan, s. d.). Le notebook est aussi utilisé comme format de support, en complément d'une vidéo. J'ai moi-même expérimenté l'apprentissage par les notebooks alors que je faisais connaissance avec les pratiques et matériaux des dispositifs algorithmiques.

Si le notebook est utilisé comme un format pédagogique, sa fonction principale est celle d'outil scientifique. Il sert à l'expérimentation et à la démonstration reproductible de nouvelles techniques et est un mode de communication entre scientifiques. Le journaliste James Somers (2018) détaille l'histoire du Jupyter Notebook dans un article suggérant qu'il remplace, peu à peu, les publications scientifiques plus traditionnelles.

Le notebook fonctionne sur le principe REPL (*read eval print loop*), ce qui signifie que le code est exécuté ligne par ligne, ou par petits fragments, plutôt que par programmes entiers. Cette approche trouve son origine dans les contraintes des débuts de l'informatique. La capacité limitée des ordinateurs ne permettait à l'époque que ce type d'exécution. Theodore Gray, alors employé par Wolfram Research inc., tourne cette contrainte en élément interactif, permettant aux scientifiques d'expérimenter pas à pas, voyant les résultats changer et affinant ainsi leur programme alors qu'il s'exécute. Si Steve Jobs lui-même a contribué aux premières interfaces, c'est Stephen Wolfram qui popularise le notebook avec son programme Mathematica (1988), ajoutant la production de graphiques et de visualisations en sortie (*output*) de chaque ligne. Si Stephen Wolfram peut être crédité pour les premières versions du notebook, cette invention est cependant enfermée dans *Mathematica*, logiciel propriétaire qui reflète la vision centralisée et totalisante de Wolfram.

En 2001, une alternative open source et collaborative au notebook Mathematica commence à être développée. L'implémentation du mode d'édition interactive dans le langage Python est nommée IPython (Pérez & Granger, 2007) et inclut un format de notebook. En 2014, Fernando Pérez (à l'origine de IPython) forme l'association Project Jupyter [4] et donne son nom au Jupyter Notebook. Selon la mythologie du mouvement open-source (Raymond, 1998), Somers (2018) décrit le notebook Mathematica comme une *cathédrale* commerciale, fermée et centralisée autour de Stephen Wolfram. Le Jupyter Notebook lui oppose un *bazar* ouvert et collaboratif, parfois au prix de devoir accommoder des points de vue divergents. Le notebook cristallise ici certaines des questions politiques autour d'une production du savoir centrée sur l'analyse de données et opérant par des outils numériques. Les visions propriétaires et open source semblent clairement démarquées et la seconde semble évidemment mieux correspondre à une recherche libre et indépendante, évitant des frais coûteux de licences aux projets de recherche financés par de l'argent public. Ces lignes deviennent plus floues lorsque l'on considère que Jupyter est développé grâce aux financements de partenaires industriels et institutionnels.

En pratique, le notebook consiste en une page contenant des cellules de code ou de texte. Chaque cellule est exécutée tour à tour, suivant ou non l'ordre linéaire de la page. Le « noyau », ou *kernel,* qui fournit l'environnement programmatique, garde en mémoire les opérations. Chaque cellule est indexée et peut produire une sortie (*output*) sous forme de texte ou de graphique. Les cellules de texte ne font pas partie du programme mais peuvent l'agrémenter d'explications, liens et images grâce à la syntaxe Markdown, ce que Donald E. Knuth (1984) appelle un mode de « programmation lettrée » (*literate programming*, traduction personnelle). Le résultat est donc un média singulier,

entre page web et programme, qui donne à voir un cheminement par le code. Un notebook publié comporte toutes les entrées (*inputs*) et sorties (*outputs*) à un moment donné, mais il peut être modifié et réexécuté par quiconque le télécharge. Cette approche d'une programmation reproductible, dynamique, visuelle et potentiellement collaborative n'est pas limitée au projet Jupyter. Nombre de formats s'en inspirent plus ou moins directement pour proposer des modes exploratoires et intuitifs d'analyse de données [5]. Vues du design, ces solutions rappellent les arguments de Bret Victor (2012) pour une programmation visuelle qui donne à voir son résultat en temps réel et encourage de nouveaux modes de création.

S'il ouvre, en théorie, des possibilités à quiconque veut s'en saisir, il est très rare de voir le notebook mobilisé à des fins critiques. Il reste un rouage parmi tant d'autres de la médiation entre données et savoir, pour rendre compte d'analyses de données, démontrer des techniques de modélisation ou communiquer de nouvelles avancées. Pendant mes explorations pour *almanac. computer,* je me suis servi du notebook pour faire, moi aussi, de l'exploration. Ce faisant, j'étais conscient que les relations que j'essayais — avec ou sans succès — d'établir entre différentes sources de données ne suivaient pas l'utilisation attendue de cet outil. Mes expérimentations mettaient en code des rationalités parallèles, divinatoires et tordaient le cou aux pratiques de la science des données (*data science*). En revenant sur cette expérience, une figure me semble illustrer la posture que j'ai prise pendant le projet : l'idiot.

L'idiot, force de « disruption »

Mon but n'est pas d'extraire une série de méthodes du projet *almanac.computer*. Ce serait répliquer les tendances généralisatrices de l'industrie de l'IA. Néanmoins, il me paraît utile de relier la pratique critique mise en œuvre pendant ce projet, en particulier l'utilisation du Jupyter Notebook, avec une posture/position du designer par rapport aux dispositifs algorithmiques. La figure de l'idiot semble répondre à ces injonctions contradictoires. Ce personnage conceptuel, théorisé par Gilles Deleuze et Félix Guattari (1991, p. 61), refuse toute rationalisation, voulant « faire de l'absurde la plus haute puissance de la pensée, c'est-à-dire créer ». L'idiot est repris par Isabelle Stengers (2007) dans une formulation qui fait écho à la pratique du design puisqu'elle est adressée aux praticiens qui « ont appris à hausser les épaules devant les prétentions des théoriciens généralisateurs. Portés à les définir comme des exécutants, chargés ‹ d'appliquer › une théorie, ou à capturer leur pratique comme illustration d'une théorie » (Stengers, 2007, p. 45).

Le travail d'*almanac.computer* consiste à questionner la validité assignée aux prédictions de la science des données et refusée à d'autres imaginaires cosmiques tels que l'astrologie. Dans les mots de Stengers, il s'agit de « rire non des théories, certes, mais de l'autorité qui leur est associée » (p. 46). Dans un moment où « l'intelligence artificielle » est promue sans relâche, tant par des vecteurs commerciaux que gouvernementaux (Villani *et al.*, 2018), l'idiot apporte un certain salut en étant « celui qui toujours ralentit les autres, celui qui résiste à la manière dont la situation est présentée, dont les urgences mobilisent la pensée ou l'action » (Stengers, 2007, p. 47).

Le « moment IA » se caractérise par une application généralisée des mêmes méthodes statistiques à tous les champs d'activité. Dans le cas d'Adobe évoqué plus haut, cette généralisation est illustrée par le fait qu'une entreprise de logiciels de création, ayant développé des capacités en intelligence artificielle, se transforme en fournisseur de services en analyse de données. Sensei est développé pour détourer des images dans Photoshop mais produit aussi des analyses de données (*data analytics*) et des analyses tendancielles de marché (*insights*) vendues telles quelles. Adobe Experience Platform propose, en outre, d'accéder à Sensei à travers des Jupyter Notebooks [6]. Face à cette poussée hégémonique de la donnée comme unique source de savoir, l'idiot, lui, « ne se ‹ résignera › jamais à ce que 3 + 2 = 5 » (Deleuze & Guattari, 1991, p. 61). Ce refus de la résignation semble être un rempart essentiel au moment où les prédictions basées sur des données *prescrivent* en permanence un futur conforme au passé.

La figure de l'idiot trouve un certain écho chez les sociologues et philosophes de la technologie, soucieux de se réapproprier la notion de spéculation. En particulier, Mike Michael (2012) rapproche l'idiot du design en l'utilisant pour analyser les pratiques spéculatives et/ou critiques. Mike Michael sépare l'idiot d'autres figures telles que le bouffon ou le saltimbanque qui se contentent de « retourner » les événements mais ne les remettent pas radicalement en cause, loin s'en faut. Sur le thème de l'intelligence artificielle, il est par exemple courant que les oppositions frontales soient utilisées comme faire-valoir par le discours dominant, brandissant l'image d'un discours « anti-progrès » pour renforcer sa position. L'idiot, pour Mike Michael, va beaucoup plus loin, son action (ou non-action) est « si incompatible avec les événements qu'elle perturbe leur interprétation orthodoxe » (p. 171, traduction personnelle). L'idiot force l'attention vers l'absurde et pousse par là une redéfinition complète du sens des événements. Suivant une méthodologie *idiotique,* le design devient une discipline pour inventer de nouveaux problèmes. Mike Michael propose *de-sign* pour qualifier cette pratique, génératrice d'ambiguïté et de trouble, qui pousse à un questionnement profond des événements et de leur interprétation.

Un exemple appliqué d'une pratique de recherche en design *idiotique* est le travail de Michael Guggenheim, Bernd Kräftner et Judith Kröll (2017). Ici, l'accent est mis sur la médiation opérée par les objets et/ou les interfaces et le mode de spéculation que ceux-ci encouragent. Les terminaux boursiers de type Bloomberg, ou autres logiciels de *trading*, réduisent le monde à des opérations vendre/acheter et forment par là des spéculateurs financiers au champ de vision et d'opération très réduits. En réponse, les chercheur·euse·s proposent de « construire des machines spéculatives qui encouragent une spéculation idiotique » (p. 146, traduction personnelle). Ces machines serviraient d'intermédiaire avec le monde dans toute sa complexité pour libérer les imaginaires, notamment face à la crise climatique. Ce mode de spéculation *cosmopolitique* est directement inspiré d'Isabelle Stengers (2007) et de sa description de l'idiot évoquée plus haut. La recherche se concentre sur une machine en particulier, conçue par les chercheurs, qui prend la forme d'un bac à sable agrémenté d'accessoires et de figurines pour explorer des mondes et des scénarios possibles dans le cadre d'un projet sur les réponses possibles aux désastres à venir. La machine de Michael Guggenheim, Bernd Kräftner et Judith Kröll (2017) comporte de l'informatique, mais uniquement pour la captation des interventions des participants. La méthode décrite est néanmoins transférable à un travail utilisant données et algorithmes comme sujet et/ou matériau. L'idiot peut être imaginé comme incapable de se servir d'un ordinateur. Cependant, de mon point de vue, il est intéressant d'imaginer quelles formes prendraient les refus et les tangentes radicales d'un idiot à l'intérieur d'un ordinateur et non de s'arrêter à son simple refus.

Je reconnais quelques éléments d'une informatique idiotique dans le projet *almanac.computer*. Positionner la prédiction algorithmique comme une pratique divinatoire, c'est chambouler les lignes de démarcation érigées avec tant d'efforts pour légitimer certains modes de spéculation (scientifique, financière) au détriment d'autres (astrologie, jeux de hasard). Le refus des « interprétations orthodoxes des événements » (Michael, 2012, p. 171,

traduction personnelle) s'opère dans le projet sur au moins deux niveaux : 1) L'histoire des almanachs met en avant l'interprétation au cœur de la production de « données ». Celles-ci ne sont pas les événements eux-mêmes — bien que souvent présentées comme telles — mais opèrent des médiations subjectives et politiques qui commencent par le choix de ce qui est ou non transformé en données. 2) Les opérations algorithmiques sont détournées, mises au service de rationalités pour lesquelles elles n'ont pas été conçues. Elles produisent cependant des résultats, démontrant ainsi l'implémentation programmatique de systèmes de croyance.

Pour une idiotique opérationnelle

La question du caractère idiotique du projet *almanac.computer* restera ici sans réponse définitive. Il serait malvenu d'apporter des conclusions fermes en utilisant la figure de l'idiot, elle qui résiste à toute théorisation. J'espère plutôt ouvrir des possibilités pour des travaux futurs, et poser les bases d'une position idiotique du design envers « l'intelligence artificielle » et autres dispositifs algorithmiques.

Le Jupyter Notebook n'est pas en lui-même un instrument critique ou idiotique. Il est au cœur des dispositifs algorithmiques. Cependant, une utilisation critique en est peut-être possible. Si l'on examine, et détourne, les médiations qu'il opère, il peut se transformer en une machine de spéculation idiotique comme celle décrite par Michael Guggenheim, Bernd Kräftner et Judith Kröll (2017). Le kernel du notebook peut, en effet, être vu comme une sorte de bac à sable ou, au moins, de terrain d'expérimentation. Ce théâtre d'opérations est propice aux situations absurdes vu qu'il garde en mémoire toutes les lignes exécutées pendant une session, qu'elles soient encore visibles à l'écran ou non. Il arrive donc qu'après de nombreux essais, le kernel se trouve dans un état totalement singulier où certaines opérations peuvent fonctionner sans pour autant être reproductibles lors de l'exécution suivante. Ajoutons à ceci les capacités narratives du format — par l'ajout de texte, d'images ou autre média —, et un potentiel s'ouvre pour détourner les outils algorithmiques, au cœur même de leur noyau opératoire.

Plus généralement, le notebook met l'accent sur une pratique et une réflexion *opérationnelles*, ce qui ne revient pas à dire qu'elles produisent des artefacts *fonctionnels* au sens « utile » du terme. Le point important étant de se saisir de l'opérationnalisation des données par les systèmes algorithmiques et de mettre en question le cœur même, le moteur, des imaginaires science-fictionnels qui sont proposés et rabâchés en permanence. Si l'idiot demande que l'absurde soit rétabli, ce n'est cependant pas forcément à lui de le produire, il peut aussi révéler l'absurde dans les systèmes existants. Cette tâche se rapproche de la « recherche d'une poétique des machines » évoquée par les médiarchéologistes (Bardini *et al.*, 2016), et du design en tant que « pratiques suffisamment engagées dans le ‹faire› des matières numériques pour y déceler des esthétiques singulières » (Masure, 2017, p. 37).

Une attention portée sur les opérations veut aussi dire, pour le design, que les images — produites ou analysées — doivent être pensées non pas en tant que surfaces lisses mais comme faisant partie d'un processus opérationnel (Farocki, 2004). Suivre les diagrammes formés par les systèmes algorithmiques mène à leurs généalogies oubliées et à leurs ramifications sociales, culturelles, et politiques. Une pratique idiotique opérationnelle positionne l'idiot au contact des dispositifs, assis à son clavier, à chercher comme tout programmeur des réponses à ses problèmes sur Stack Overflow. Cette position n'est définitivement pas le simple « retournement » du saltimbanque (Michael, 2012) ou autre refus en bloc de l'algorithmique ou de l'informatique en général. Comme l'indiquent les « statactivistes » Emmanuel Didier et Cyprien Tasset (2013), refuser serait laisser « le monopole de ces instruments aux puissants. Il n'y a pas de raison pour que la quantification se trouve toujours du côté de l'État et du capital » (p. 124).

Le design, à mon sens, se doit donc de *faire l'idiot avec* les dispositifs algorithmiques. C'est-à-dire être à la fois au contact de leurs pratiques et opérations et se tenir à distance de leur rationalité. S'approprier la mécanique mais remettre en cause les récits hégémoniques, par exemple, la montée déjà bien avancée du positivisme numérique. Insister en demandant, comme l'idiot : « un problème plus profond, quel problème ? Je ne vois pas bien, mais laissez-moi, laissez-moi » (Deleuze, 1987). Gilles Deleuze met en avant la relation indissociable entre les idées et les médias dans lesquels elles sont exprimées. Ce sont, selon lui, « des potentiels déjà engagés dans tel ou tel mode d'expression ». Il y a des idées *en* cinéma, *en* peinture, *etc.* Des croisements peuvent toutefois s'opérer, par exemple la figure de l'idiot tisse des liens entre des idées *en* roman (Dostoïevski), des idées *en* cinéma (Kurosawa) et des idées *en* philosophie (Deleuze). Le potentiel narratif et opérationnel du Jupyter Notebook est un terrain pour avoir des idées *en* algorithmes. Ce qui est démontré par ses utilisations pédagogiques et scientifiques. Il est aussi propice à une réappropriation par un design sensible aux matériaux (dans ce cas : données, code, *outputs*, visualisations, etc.) et aux forces culturelles, sociales et politiques qu'ils manifestent. Sonder, manipuler et contester ces matériaux et ces forces est, selon moi, la contribution que le design peut apporter au « moment IA » actuel. Pour reprendre la métaphore mythique du mouvement open-source (Raymond, 1998), un design critique face à la cathédrale de l'IA se doit de mettre le bazar.

NOTES

1 À consulter sur https://almanac.computer et https://davidbenque.com/projects/almanac.

2 *Datalmanach*, mené durant la semaine de recherche *S47* du 21 au 24 novembre 2016 : www.esadse.fr/fr/s47/211016-workshops.

3 Les outils liés au langage Python, comme Pandas pour la manipulation de données, Scikit-learn pour le machine-learning, et Matplotlib pour la visualisation.

4 À consulter sur jupyter.org. Jupyter diverge de IPython en ceci que l'association gère les éléments qui ne sont pas liés à un langage de programmation particulier. Le Jupyter Notebook peut être utilisé avec n'importe quel langage. *IPython* désigne toujours le kernel Python compatible avec le notebook.

5 Par exemple, Markdown, Rmarkdown. rstudio.com, Observable observablehq.com, ou Google Colaboratory, colab.research.google.com.

6 Consultables en ligne, Docs.adobe.com/content/help/en/experience-platform/data-science-workspace/jupyterlab/overview.html.

Références

Adobe (2020), *Sensei: creative-cloud-artificial-intelligence*. Consulté le 26 septembre 2020, sur https://www.adobe.com/fr/sensei/creative-cloud-artificial-intelligence.html

AGRAWAL, Ajay, GANS, Joshua et GOLDFARB, Avi, *Prediction machines: The simple economics of artificial intelligence*, Cambridge, Harvard Business Review Press, 2018.

BARDINI, Thierry, BROYE, Lionel, CITTON, Yves., GALLIGO, Igor, GUEZ, Emmanuel, GUESS, Jeff, JULIEN, Quentin, KRZYWKOWSKI, Isabelle, LECHNER, Marie, MASURE, Anthony., PANDELAKIS, Saul, THIBAULT, Ghislain et VARGOZ Frédérique, *Manifeste Médiarchéologiste*, Avignon, PAMAL (Preservation & Art – Media Archaeology Lab) – École Supérieure d'Art d'Avignon, 2016.

http://wiki.pamal.org/wiki/Manifeste_M%C3%A9diarch%C3%A9ologiste

BENQUÉ, David, « Cosmic Spreadsheets », dans VOSS Georgina (dir.), *Supra Systems #1*. Londres, London College of Communication (University of the Arts London), 2018. http://suprasystems.studio/downloads/book-chapters/Supra%20Systems%20Book_Chapter%2010_Benque.pdf

BENQUÉ, David, *Case board, traces, & chicanes: Diagrams for an archaeology of algorithmic prediction through critical design practice* [Thèse de doctorat], Londres, Royal College of Art, 2020. https://researchonline.rca.ac.uk/4420/

DAVIDSON-PILON, Cameron, *Probabilistic Programming & Bayesian Methods for Hackers*, Addison Wesley, 2015. http://camdavidsonpilon.github.io/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers/

DELEUZE, Gilles, « Qu'est-ce que l'acte de création ? » [Conférence enregistrée], Mardis de la fondation Fémis (école nationale supérieure des métiers de l'image et du son), Paris, 17 mai 1987.

DELEUZE, Gilles et GUATTARI, Félix, *Qu'est-ce que la philosophie*, Éditions de Minuit, 1991.

DIDER, Emmanuel, et TASSET, Cyprien (2013). Pour un statactivisme. La quantification comme instrument d'ouverture du possible. *Tracés*, *24*, 123-140. https://doi.org/10.4000/traces.5660

DOWNEY, Allen (2020). *Elements of Data Science* [Jupyter Notebook]. https://github.com/AllenDowney/ElementsOfDataScience

FAROCKI, Harun (2004). Phantom Images. *Public*, *0*(29), Article 29. https://public.journals.yorku.ca/index.php/public/article/view/30354

Google Arts & Culture, & Google AI. (2019, mai). *Artists + Machine Intelligence Grants*. Experiments with Google. https://experiments.withgoogle.com/ami-grants

GUGGENHEIM, Michael, KRÄFTNER, Bernd, & KRÖLL, Judith (2017). Creating idiotic speculators; Disaster cosmopolitics in the sandbox. In Alex WILKIE, Martin SAVRANSKY, & Marsha ROSENGARTEN (Éds.), *Speculative Research* (p. 145-162). Routledge.

JURGENSON, Nathan (2014, octobre 9). View From Nowhere. *The New Inquiry*. https://thenewinquiry.com/view-from-nowhere/

KNUTH, Donald E. (1984). Literate Programming. *The Computer Journal*, *27*(2), 97-111. https://doi.org/10.1093/comjnl/27.2.97

KOGAN, Gene (s. d.). *Guides*. Machine Learning for Artists (ml4a). Consulté 2 octobre 2020, à l'adresse https://ml4a.github.io/guides/

LAFRANCE, Adrienne (2015, novembre 13). How The Old Farmer's Almanac Previewed the Information Age. *The Atlantic*. https://www.theatlantic.com/technology/archive/2015/11/how-the-old-farmers-almanac-previewed-the-information-age/415836/

MASURE, Anthony (2014). *Le design des programmes* [Thèse de doctorat], Université Paris1 Panthéon-Sorbonne. http://www.softphd.com/

MASURE, Anthony (2017). *Design et humanités numériques*. Éditions B42.

MCQUILLAN, Dan (2017). Data Science as Machinic Neoplatonism. *Philosophy and Technology*, *3*(1), 253-272. https://doi.org/10.1007/s13347-017-0273-3

MICHAEL, Mike (2012). De-Signing the Object of Sociology: Toward an 'Idiotic' Methodology. *The Sociological Review*, *60*(S1), 166–183. https://doi.org/10.1111/j.1467-954X.2012.02122.x

PÉREZ, Fernando, & GRANGER, Brian E. (2007). I Python: A System for Interactive Scientific Computing. *Computing in Science and Engineering*, *9*(3), 21-29. https://doi.org/10.1109/MCSE.2007.53

RAYMOND, Eric Steven (1998) *La cathédrale et le bazar (The Cathedral and the Bazaar)*. Traduit par S. BLONDEEL, http://www.linux-france.org/article/these/cathedrale-bazar/cathedrale-bazar.html

SHAPIRO, Aaron (2015, avril). The Next Big Thing In Design ? Less Choice. *fastcodesign.com*. https://www.fastcodesign.com/3045039/the-next-big-thing-in-design-fewer-choices

SOMERS, James (2018, avril 5). The Scientific Paper Is Obsolete. *The Atlantic*. https://www.theatlantic.com/science/archive/2018/04/the-scientific-paper-is-obsolete/556676/

STENGERS, Isabelle (2007). La proposition cosmopolitique. In J. Lolive & O. Soubeyran, *L'émergence des cosmopolitiques* (p. 45-68). La Découverte.

VICTOR, Bret (2012). *Inventing on Principle*. Canadian University Software Engineering Conference. https://vimeo.com/36579366

VILLANI, Cédric, SCHOENAUER, Marc, BONNET, Yann, BERTHET, Charly, CORNUT, Anne-Charlotte, LEVIN, François, & RONDEPIERRE, Bertrand (2018). *Donner un sens à l'intelligence artificielle*. Conseil national du numérique. https://hal.inria.fr/hal-01967551

Cosmic Commodity Charts

Predicting prices of commodity futures according to the positions of solar system planets.

Stock market data: Quandl
Astronomical data: NASA Jet Propulsion Laboratory de430 ephemeris [PDF]
accessed via Skyfield

In [1]:
```
import numpy as np
import pandas as pd
import datetime
import matplotlib.pyplot as plt
%matplotlib inline
```

Chart title and query

In [2]:
```
# all commodity options

commodities = {
    # Grain and foods
    "Corn": "CHRIS/CME_C1",
    "Wheat": "CHRIS/CME_W3",
    "Soybean": "CHRIS/CME_S3",
    "Rough Rice": "CHRIS/CME_RR1", # only goes back t
    "Cotton No.2": "CHRIS/ICE_CT2",
    "Sugar": "CHRIS/ICE_SB2",
    "Orange Juice": "CHRIS/ICE_OJ4",
    "Coffee C": "CHRIS/ICE_KC3",
    # Livestock
    "Live Cattle": "CHRIS/CME_LC3",
    "Feeder Cattle": "CHRIS/CME_FC2",
    "Lean Hog": "CHRIS/CME_LN4",
    # energy
    "Crude Oil": "CHRIS/CME_CL38",
    # Metals
    "Copper": "CHRIS/CME_HG2",
    # Precious
    "Gold": "CHRIS/CME_GC5",
    "Silver": "CHRIS/CME_SI7",
    }
```

In [3]:
```
# planet options

planet_lists = {
    "Full": ['mercury', 'venus', 'earth', 'mars',
             'jupiter', 'saturn', 'uranus', 'neptune',
    "Inner": ['mercury', 'venus', 'earth', 'mars']
}
```

In [4]:
```
# chart lengths in days

lengths = [365, 730]
```

In [5]:
```
# Build the query

from random import choice

plans = list(planet_lists.keys())
coms = list(commodities.keys())

query = [choice(plans), choice(coms), choice(lengths)
print("{0} {1} Chart, {2} Days".format(*query))
```

```
Inner Copper Chart, 365 Days
```

Commodity Price

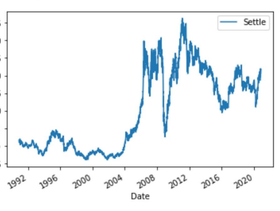
In [6]:
```
import quandl

with open("../_data/quandl-key.txt", "r") as text:
    for line in text:
        quandl_key = line
quandl.ApiConfig.api_key = quandl_key
```

In [7]:
```
today = datetime.date.today()
start_date = today - datetime.timedelta(days = 10950)

price_data_30y = quandl.get(commodities[query[1]], st
price_data_30y.head()
```

Out[7]:

Date	Open	High	Low	Last	Change	Settle	Volume	Previous Day Open Interest
1990-11-26	1.152	1.168	1.1490	NaN	NaN	1.1645	2712.0	13217.0
1990-11-27	1.157	1.158	1.1220	NaN	NaN	1.1230	6530.0	11982.0
1990-11-28	1.123	1.128	1.1050	NaN	NaN	1.1225	5226.0	11080.0
1990-11-29	1.126	1.130	1.1100	NaN	NaN	1.1120	3329.0	9631.0
1990-11-30	1.118	1.127	1.1045	NaN	NaN	1.1110	3154.0	8073.0

In [8]:
```
price_data_30y.plot(y="Settle")
```

Out[8]:


Planets & Orbits Training Data

In [9]:
```
# load skyfield and ephemeris
from skyfield.api import load, Loader
from astropy import units as u

# Skyfield data
load = Loader('../_data/skyfield')
planets = load('de430.bsp')
ts = load.timescale()
```

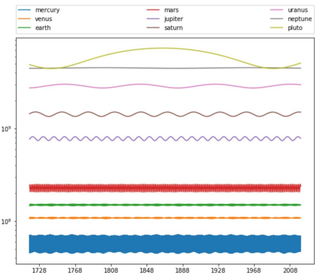
In [10]:
```
planet_lists = {
    "Full": ['mercury', 'venus', 'earth', 'mars',
             'jupiter', 'saturn', 'uranus', 'neptune',
    "Inner": ['mercury', 'venus', 'earth', 'mars'],
    "Outer": ['jupiter', 'saturn', 'uranus', 'neptune
}
```

In [11]:
```
planet_list = planet_lists[query[0]]
```

In [12]:
```
def add_planet_positions(df, planet_list):
    '''
    input: dataframe with datetime index, list of pla
    output: dataframe with planet distance (barycent
            added for the planets in planet_list and
    '''
    for planet in planet_list:
        query_name = planet + " barycenter"
        col_name = planet + "_dist"
        p = planets[query_name]
        df[col_name] = p.at(ts.utc(df.index)).distanc
    return df
```

Scaling

The planet positions are scaled according to their relative orbits, with 0.0 and 1.0 as the closest and furthest points from the solar system barycentre on that planet's orbit.



In [13]:
```
# Mininum and maximum for each orbit
# 300 year window to allow for a full orbit of Pluto
# exported to a CSV file to avoid computing them each
planets_minmax = pd.read_csv("planets_minmax.csv", in
planets_minmax
```

In [13]:

	min	max
mercury_dist	4.473931e+07	7.108315e+07
venus_dist	1.062104e+08	1.102090e+08
earth_dist	1.458724e+08	1.533281e+08
mars_dist	2.052792e+08	2.505484e+08
jupiter_dist	7.390777e+08	8.158063e+08
saturn_dist	1.346026e+09	1.508435e+09
uranus_dist	2.735045e+09	3.006735e+09
neptune_dist	4.459924e+09	4.536853e+09
pluto_dist	4.435661e+09	7.375996e+09

In [14]:
```
def orbit_scaler(planet_dist, x):
    min_dist, max_dist = planets_minmax.loc[planet_d
    return ((x - min_dist)/(max_dist-min_dist))
```

In [15]:
```
training_data = price_data_30y["Settle"].to_frame()
training_data["Date"] = training_data.index

import pytz
from pytz import timezone

utc = timezone('UTC')
def to_utc(x):
    # add utc timezone to datetime index
    a = x.replace(tzinfo=pytz.utc)
    return a

training_data["Date"] = training_data["Date"].apply(t
training_data.set_index(['Date'], inplace=True)
training_data = training_data.rename(columns={"Settle
training_data = add_planet_positions(training_data, p
training_data.head()
```

In [15]:

Date	Price	mercury_dist	venus_dist	earth_dist
1990-11-26 00:00:00+00:00	1.1645	6.326097e+07	1.085074e+08	1.478144e+08
1990-11-27 00:00:00+00:00	1.1230	6.256071e+07	1.085266e+08	1.477865e+08
1990-11-28 00:00:00+00:00	1.1225	6.183206e+07	1.085456e+08	1.477592e+08
1990-11-29 00:00:00+00:00	1.1120	6.107705e+07	1.085643e+08	1.477324e+08
1990-11-30 00:00:00+00:00	1.1110	6.029799e+07	1.085828e+08	1.477061e+08

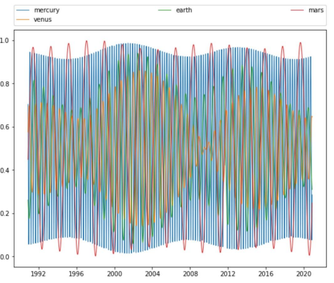
In [16]:
```
target = training_data["Price"]
features = training_data.drop(["Price"], axis=1)
```

In [17]:
```
planet_distances = plt.figure(figsize=(9,7))
ax = plt.subplot()
for c in list(features):
    line = ax.plot(features.index, features[c], label
ax.set_yscale('log')
ax.legend(bbox_to_anchor=(0., 1.02, 1., .102), loc=3,
          ncol=3, mode="expand", borderaxespad=0.)
```

In [17]:

In [18]:
```
features_scaled = pd.DataFrame()
for column in features:
    features_scaled[column] = orbit_scaler(column, fe

planet_distances_scaled = plt.figure(figsize=(9,7))
ax = plt.subplot()
for c in list(features_scaled):
    line = ax.plot(features_scaled.index, features_sc
ax.legend(bbox_to_anchor=(0., 1.02, 1., .102), loc=3,
          ncol=3, mode="expand", borderaxespad=0.)
```

Out[18]: <matplotlib.legend.Legend at 0x7fe8b2b4bc10>



Training

In [19]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    features_scaled.values, target.values, test_size
```

In [20]:
```
from sklearn import svm

clf = svm.SVR(kernel='rbf', gamma="auto")
clf.fit(X_train, y_train)
clf.score(X_test, y_test)
```

Out[20]: -0.032653302362780856

Prediction

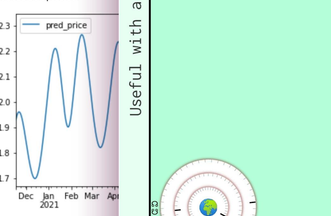
In [21]:
```
tomorrow = datetime.date.today() + datetime.timedelta
end_date = tomorrow + datetime.timedelta(days=query[1
dates = pd.date_range(tomorrow, end_date, freq = 'D'
dates = pd.to_datetime(dates)
future = pd.DataFrame(index=dates, columns=['A'])
future = add_planet_positions(future, planet_list)
future.drop(["A"], axis=1, inplace=True)

# scale
for column in futur
    future[column]

# predict price
future["pred_price"

# export
future["pred_price"

future.head()
```

Out[21]:

	mercury_
2020-11-17 00:00:00+00:00	0.406
2020-11-18 00:00:00+00:00	0.437
2020-11-19 00:00:00+00:00	0.468
2020-11-20 00:00:00+00:00	0.499
2020-11-21 00:00:00+00:00	0.529

In [22]:
```
future.plot(y="pred
```

Out[22]: <AxesSubplot:>



In [23]:
```
future["change"] =
price_preds = futur
price_preds = price
price_preds['direct
price_preds['dir_ch

# remove change flag for first and last dates
for date in [0, -1]:
    price_preds.iat[date, 3] = 0 # set dir_change to

price_preds["block"] = price_preds.dir_change.cumsum
price_preds.head()
```

Out[23]:

	pred_price	change	direction	dir_change	block
2020-11-17 00:00:00+00:00	1.931283	-0.013012	-1.0	0	0
2020-11-18 00:00:00+00:00	1.944294	-0.009280	-1.0	0	0
2020-11-19 00:00:00+00:00	1.953574	-0.005561	-1.0	0	0
2020-11-20 00:00:00+00:00	1.959135	-0.001938	-1.0	0	0
2020-11-21 00:00:00+00:00	1.961073	0.001517	1.0	1	1

In [24]:
```
# Format and Output peaks and valleys data

# remove weird warning for now
pd.options.mode.chained_assignment = None

#select rows with a direction change
peaks_valleys = price_preds.loc[price_preds["dir_char

#dump unncessary columns
peaks_valleys.drop(["change","dir_change", "block"],

# make a date column, format and dump the index
peaks_valleys["date"] = peaks_valleys.index
#peaks_valleys["date"] = peaks_valleys["date"].apply(
peaks_valleys = peaks_valleys.reset_index(drop=True)

# truncate price to 2 decimals
#peaks_valleys["pred_price"] = peaks_valleys["pred_pr

# export csv wihtout the index column
peaks_valleys.to_csv("_temp/peaks_valleys.csv", index
peaks_valleys
```

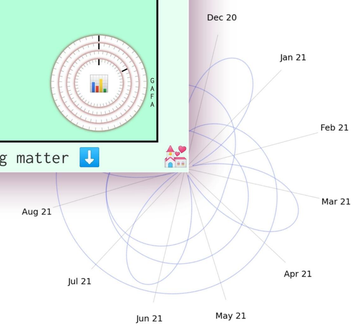
Out[24]:

	pred_price	direction	date
0	1.961073	1.0	2020-11-21 00:00:00+00:00
1	1.698898	-1.0	2020-12-13 00:00:00+00:00
2	2.210239	1.0	2021-01-10 00:00:00+00:00
3	1.901611	-1.0	2021-01-27 00:00:00+00:00
4	2.264358	1.0	2021-02-15 00:00:00+00:00
5	1.820753	-1.0	2021-03-13 00:00:00+00:00
6	2.236116	1.0	2021-04-06 00:00:00+00:00
7	1.803543	-1.0	2021-04-26 00:00:00+00:00
8	2.159872	1.0	2021-05-14 00:00:00+00:00
9	1.726347	-1.0	2021-06-09 00:00:00+00:00
10	2.093337	1.0	2021-07-03 00:00:00+00:00
11	1.730149	-1.0	2021-07-22 00:00:00+00:00
12	2.315678	1.0	2021-08-12 00:00:00+00:00
13	1.958547	-1.0	2021-09-05 00:00:00+00:00
14	2.258558	1.0	2021-09-28 00:00:00+00:00
15	1.732678	-1.0	2021-10-19 00:00:00+00:00
16	2.007736	1.0	2021-11-05 00:00:00+00:00

Charting layers

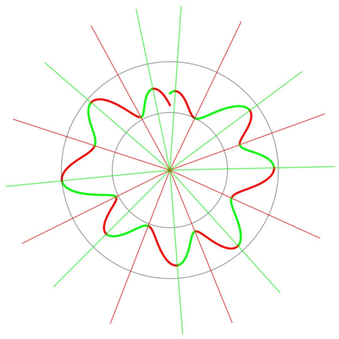
In [25]:
```
import matplotlib.dates as mdates
import matplotlib.ticker as ticker
```

In [26]:
```
# Features (planets) chart

# months for axis ticks - fill month_starts and drop
future["month"] = future.index.strftime('%m').astype
future["month_diff"] = future.month.diff().fillna(0)
month_starts = future.loc[future['month_diff'] != 0]
future.drop(["month","month_diff"], axis=1,inplace=Tr

# remove backgrounds (has to be specified again for s
plt.rcParams['figure.facecolor'] = ((0,0,0,0.))
plt.rcParams['axes.facecolor'] = ((0,0,0,0.))

# create plot
```



In [27]:
```
# create plot
fig = plt.figure(figsize=(15,15))
ax = plt.subplot(projection='polar')
ax.set_theta_direction(-1)
ax.set_theta_zero_location("N")

# normalise theta axis and ticks
t = mdates.date2num(future.index.to_pydatetime())
tnorm = (t-t.min())/(t.max()-t.min())*2.*np.pi

# set min and max
import math
price_min = future["pred_price"].min()
axis_min = math.floor(price_min)

price_max = future["pred_price"].max()
axis_max = math.ceil(price_max)
ax.set_ylim(axis_min, axis_max)

peak_ticks= []
peak_labels = []
peak_values = []

for index, row in peaks_valleys.iterrows():
    m = mdates.date2num(row['date'].to_pydatetime())
    mnorm = (m-t.min())/(t.max()-t.min())*2.*np.pi
    #peak_ticks.append(mnorm)
    peak_labels.append("${0:.2f} ".format(row['pred_p
    peak_values.append(row['pred_price'])
    if row['direction'] < 0:
        c = 'r'
    else:
        c = (0,1,0)
    ax.plot([axis_min,mnorm],[axis_min,axis_max], lin
```

Out[24]:



Combine layers into the final chart

In [28]:
```
from PIL import Image, ImageDraw, ImageFont
import os
```

In [29]:
```
# dim of the matplotlib PDFs
width = 1080
height = 1400

center_x = width/2
center_y = (width/2) + (height - width)
```

In [30]:
```
preds = pd.read_csv('_temp/pred_price.csv', names =
first_date = preds.iloc[0]["date"].split(" ")[0]
first_day = datetime.datetime.strptime(first_date, "%
first_price = round(float(preds.iloc[0]["price"]), 2

last_date = preds.iloc[-1]["date"].split(" ")[0]
last_day = datetime.datetime.strptime(last_date, "%Y-
last_price = round(float(preds.iloc[-1]["price"]), 2
```

In [31]:
```
def price_flag(date, price, side, up=None, diff=None)
    '''
    draw price boxes at the top
    date: a datetime
    price: float
    side: "L" or "R"
    up: Boolean
    diff: price difference float
    '''

    w = 400
    h = 140
    y = ((height - width)/2) - (h/2)

    if side == "L":
        x = center_x - w
    else:
        x = center_x
    if side == "L":
        if up == True:
            y -= 30
        else:
            y += 30

    if up == True:
        color = (0,255,0) # green
    elif up == False:
        color = (255, 0, 0) # red
    else:
        color = 'black'


    idraw.rectangle((x, y, x + w, y + h), fill=No
    date_string = date.strftime("%Y-%m-%d")
    idraw.text((x + 10, y + 5), date_string, fil

    sign = ""
    if diff:
        if diff > 0:
            sign = "+"
        if diff < 0:
            sign = "-"
        price_diff = sign + "$" + str(round(abs(
        idraw.text((x + 10, y + 150), price_diff

    price_tag = "$" + str(price)
    idraw.text((x + 10, y + 30), price_tag, fill=
```

In [32]:
```
# %% fonts (proprietary, not included in the reposit
font_path = os.environ['TMA_home'] + '/_data/fonts/'
operator_date = ImageFont.truetype(font= font_path +
operator_b_diff = ImageFont.truetype(font= font_path
operator_b_price = ImageFont.truetype(font= font_path
calibri = ImageFont.truetype(font= font_path + "Calib
```

In [33]:
```
# %% set up image
img = Image.new('RGBA', (width,height))
idraw = ImageDraw.Draw(img)

#background
idraw.rectangle((0,0,width,height), fill="white")


# %% Matplotlib Layers

planets_lyr = Image.open("_temp/planets.png")
preds_lyr = Image.open("_temp/pred_price.png")

for layer in [planets_lyr, preds_lyr]:
    #layer = layer.thumbnail((1080,1080), Image.ANTIA
    img.paste(layer, (-12,height-width), layer)


# %% Price Labels and center line

price_flag(first_day, first_price, "R")

is_up = first_price < last_price
price_flag(last_day, last_price, "L", up=is_up, diff=

# center line and arrow
up_offset = 60 if is_up else 90
idraw.line(((center_x, (height-width)+40),(center_x,(
idraw.line(((center_x, center_y),(center_x,430)], fi

idraw.text((center_x-5,(height-width)-60), "→", fill=

# %% description
chart_info = open("_temp/CCC_title.txt").readlines()
chart_title = chart_info[4].rstrip()

idraw.text((10,height-50), chart_title, font=operator

# %% min and max
minmax = "{type}: ${price} - {date}"
min = preds.loc[preds['price'].idxmin()]
max = preds.loc[preds['price'].idxmax()]

min_str = minmax.format(type="MIN", price=round(min.p
max_str = minmax.format(type="MAX", price=round(max.p

idraw.text((10,5), min_str + "    " + max_str, font=op
```

In [34]:
```
print(f"{query[0]} {query[1]} Chart")
print(f"{query[2]} Days, starting {chart_info[0]}")
print(f"Futures contract: {chart_info[4]}")
img
```

```
Inner Copper Chart
365 Days, starting 2020-10-30 00:10:03.520291

Futures contract: CHRIS/CME_RR1
```

Out[34]:
```
MIN: $1.7 - 2020-12-13    MAX: $2.32 - 2021-08-12
```


Buying/Selling opportunities

In [35]:
```
# Buying Low
peaks_valleys.loc[peaks_valleys['direction'] == -1]
```

Out[35]:

	pred_price	direction	date
1	1.698898	-1.0	2020-12-13 00:00:00+00:00
3	1.901611	-1.0	2021-01-27 00:00:00+00:00
5	1.820753	-1.0	2021-03-13 00:00:00+00:00
7	1.803543	-1.0	2021-04-26 00:00:00+00:00
9	1.726347	-1.0	2021-06-09 00:00:00+00:00
11	1.730149	-1.0	2021-07-22 00:00:00+00:00
13	1.958547	-1.0	2021-09-05 00:00:00+00:00
15	1.732678	-1.0	2021-10-19 00:00:00+00:00

In [36]:
```
# Selling High
peaks_valleys.loc[peaks_valleys['direction'] == 1]
```

Out[36]:

	pred_price	direction	date
0	1.961073	1.0	2020-11-21 00:00:00+00:00
2	2.210239	1.0	2021-01-10 00:00:00+00:00
4	2.264358	1.0	2021-02-15 00:00:00+00:00
6	2.236116	1.0	2021-04-06 00:00:00+00:00
8	2.159872	1.0	2021-05-14 00:00:00+00:00
10	2.093337	1.0	2021-07-03 00:00:00+00:00
12	2.315678	1.0	2021-08-12 00:00:00+00:00
14	2.258558	1.0	2021-09-28 00:00:00+00:00
16	2.007736	1.0	2021-11-05 00:00:00+00:00